

國立交通大學

資訊科學與工程研究所

碩士論文

利用有視覺自動車作室內安全巡邏與危險  
情況之偵測



Security Patrolling and Danger Condition Monitoring in  
Indoor Environments by Vision-based Autonomous  
Vehicle Navigation

研究生：江凱立

指導教授：蔡文祥 教授

中華民國 九十五年六月

利用有視覺自動車作室內安全巡邏與危險情況之偵測  
Security Patrolling and Danger Condition Monitoring in Indoor  
Environments by Vision-based Autonomous Vehicle Navigation

研究生：江凱立

Student : Kai-Li Chiang

指導教授：蔡文祥

Advisor : Wen-Hsiang Tsai

國立交通大學  
資訊科學與工程研究所  
碩士論文



Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

In

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

# 利用有視覺自動車作室內安全巡邏與危險情況之偵測

研究生：江凱立

指導教授：蔡文祥 博士

國立交通大學資訊科學與工程研究所

## 摘要



本研究主要是提出一套基於電腦視覺技術，讓自動車航行在室內環境中具有偵測危險情況與監控繪畫安全之能力。我們利用一台小型自動車作為實驗平台，並且利用無線操控的方式讓自動車航行在室內的環境中。我們提出了基於房屋中天花板角落的自動車定位方式，根據天花板角落的線條形成情況我們可以計算自動車在地圖上的實際位置。我們持續觀察環境影像並統計環境中的顏色特徵與連續影像的顏色變化，來判斷目前的環境情況是否安全。依據環境中的顏色統計資訊與變化情況，我們可以偵測出火災發生或環境停電。我們利用觀察影像中的限定區域，判斷是否遇到障礙物，以順利避開障礙物繼續巡邏。對於牆壁上的掛畫，我們提出一套將歪斜影像修復的方法來得到正確的掛畫影像，並基於此修正過的掛畫影像進行保全掛畫之安全巡邏。最後我們以成功的定位和偵測環境實驗結果證明本系統的完整性與可行性。

# **Security Patrolling and Danger Condition Monitoring in Indoor Environments by Vision-based Autonomous Vehicle Navigation**

Student: Kai-Li Chiang

Advisor: Wen-Hsiang Tsai

Institute of Computer Science and Engineering  
National Chiao Tung University

## **ABSTRACT**

A vision-based vehicle system for security patrolling and danger condition monitoring in indoor environments using an autonomous vehicle is proposed. A small vehicle with wireless control and image grabbing capabilities is used as a test bed. A camera with panning, tilting, and zooming capabilities is used as the eye of the small vehicle. A vehicle location estimation method using house corners is proposed first. This is a vision-based estimation method to prevent the mechanic error accumulation in navigation. Next, several danger condition detection methods are proposed. We propose a method to change the path for obstacle avoidance in navigation using a map of nodes. To detect fire and lighting failure conditions, we monitor the image continuously and analyze the color feature in the HSI color model. We have also proposed a method for security monitoring of paintings on walls, including techniques of painting image rectification and matching. Good experimental results show flexibility and feasibility of the proposed methods for the application of indoor security patrolling.

# ACKNOWLEDGEMENTS

I am in hearty appreciation of the continuous guidance, discussions, support, and encouragement received from my advisor, Dr. Wen-Hsiang Tsai, not only in the development of this thesis, but also in every aspect of my personal growth.

Thanks are due to Mr. Tsung-Yuan Liu, Mr. Chih-Jen Wu, Miss. Pei-Pei Chen, Mr. Yen-Long Chen, Mr. Kuo-Feng Chien, Miss Yu-Tzu Wang, Miss Chia-Yu Hsu, and Miss Yi-Lin Wang for their valuable discussions, suggestions, and encouragement. Appreciation is also given to the colleagues of the Computer Vision Laboratory in the Institute of Computer Science and Engineering at National Chiao Tung University for their suggestions and help during my thesis study.

Finally, I also extend my profound thanks to my family for their lasting love, care, and encouragement. I dedicate this dissertation to my beloved parents and girlfriend.

# CONTENTS

<b>ABSTRACT (in Chinese)</b> .....	<b>I</b>
<b>ABSTRACT (in English)</b> .....	<b>II</b>
<b>ACKNOWLEDGEMENTS</b> .....	<b>III</b>
<b>CONTENTS</b> .....	<b>IV</b>
<b>LIST OF FIGURES</b> .....	<b>VII</b>
<b>Chapter 1 Introduction</b> .....	<b>1</b>
1.1 Motivation .....	1
1.2 Survey of Related Studies .....	2
1.3 Overview of Proposed Approach .....	3
1.4 Contributions .....	5
1.5 Thesis Organization .....	6
<b>Chapter 2 System Configuration and Navigation</b>	
<b>Principles</b> .....	<b>7</b>
2.1 Overview .....	7
2.2 Advantage of Using Pan-tilt-zoom IP Cameras .....	8
2.3 System Configuration .....	9
2.3.1 Hardware Configuration .....	9
2.3.2 Software Configuration .....	13
2.4 Vehicle Guidance Principle .....	13
<b>Chapter 3 Vehicle Guidance by Vehicle Location</b>	
<b>Estimation Using A House Corner</b>	
<b>Detection Technique</b> .....	<b>16</b>
3.1 Overview .....	16

3.2	Review of Vehicle Location Estimation Techniques .....	17
3.3	Vehicle Location Estimation by House Corner Detection .....	19
3.3.1	Coordinate Systems .....	20
3.3.2	Principle of Proposed Method .....	21
3.3.3	Location Estimation by Coefficients of Edge Line Equations ...	22
3.4	Image Processing for Vehicle Location Estimation .....	26
3.4.1	Detection of House Corner Edges .....	27
3.4.2	Constrained Line Refitting and Determination of Non-vertical House Corner Edges .....	31

## **Chapter 4 Obstacle Avoidance by Goal-Directed**

### **Minimum-Path Following .....40**

4.1	Overview of Obstacle Avoidance .....	40
4.1.1	Coordinate Systems .....	42
4.1.2	Direction Angle of Vehicle .....	43
4.1.3	Coordinate Transformation .....	44
4.2	Obstacle Detection Process .....	44
4.2.1	Finding Floor Area .....	46
4.2.2	Determination of Obstacle Existence .....	47
4.3	Minimum-Path Following Process .....	49
4.3.1	Detection of Obstacle Distribution .....	49
4.3.2	Computation of Goal-Direction Minimum Path .....	51

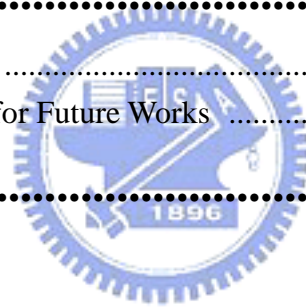
## **Chapter 5 Vision-Based Indoor Danger Condition**

### **Monitoring .....56**

5.1	Principle of Danger Condition Monitoring .....	56
5.2	Advantage of Using HSI Color Model .....	57
5.3	Dangerous Fire Condition .....	59
5.3.1	Fire Image Features .....	59
5.3.2	Proposed Fire Detection Method .....	60
5.4	Lighting Failure Condition .....	64
5.4.1	Lighting Failure Features .....	64
5.4.2	Process for Lighting Failure Detection .....	65

## **Chapter 6 Security Monitoring of Paintings on Walls in Indoor Environments by Image**

	<b>Rectification Technique .....</b>	<b>70</b>
6.1	Overview .....	70
6.2	Rectification of Images of Paintings on Walls .....	71
6.2.1	Difference Between Images of Paintings on Walls And Objects on Floors .....	71
6.2.2	Rectification of Skewed Images .....	72
6.3	Security Monitoring of Paintings on Walls by Image Analysis .....	75
6.3.1	Painting Image Features .....	75
6.3.2	Learning Process .....	76
6.3.3	Painting Security Monitoring Process .....	82
<b>Chapter 7 Experimental Results and Discussion ....</b>		<b>89</b>
<b>Chapter 8 Conclusions and Suggestions for Future Works .....</b>		<b>94</b>
8.1	Conclusions .....	94
8.2	Suggestions for Future Works .....	95
<b>References .....</b>		<b>96</b>





# LIST OF FIGURES

Figure 1.1	Flowchart of proposed system.....	5
Figure 2.1	Action of a pan-tilt-zoom camera.....	9
Figure 2.2	Structure of the system .....	10
Figure 2.3	The vehicle used in this study. (a) The bevel view of the vehicle. (b) The front view of the vehicle. (c) The left side view of the vehicle. ....	11
Figure 2.4	The pan-tilt-zoom camera used in this study. (a) The bevel view of the camera. (b) The front view of the camera. (c) The right side view of the camera.....	12
Figure 2.5	Flowchart of proposed navigation process .....	14
Figure 3.1	Three different landmarks. (a) A diamond-shaped standard mark used in [6]. (b) A sphere used for robot location in [7]. (c) The perspective projection of a house corner used in Chou and Tsai [8].....	18
Figure 3.2	The two red lines of the perspective projection of a house corner used in the study. ....	20
Figure 3.3	The three coordinate systems used in this study.....	21
Figure 3.4	Flowchart of vehicle location estimation. ....	27
Figure 3.5	Flowchart of house corner edges detection .....	27
Figure 3.6	(a) The color corner images. (b) The Canny result images of (a). ....	28
Figure 3.7	The image after applying Hough transform in Figure 3.6 (b). ....	29
Figure 3.8	An experimental result of a different house corner. (a) Another corner image. (b) The result of applying Canny edge detection of (a). The result of applying Hough transform of (b).....	30
Figure 3.9	One more experimental result of another different house corner. (a) Another corner image. (b) The result of applying Canny edge detection of (a). The result of applying Hough transform of (b). ....	31
Figure 3.10	Flowchart of the coefficients of corner edge equation computation. ....	32
Figure 3.11	Illustration of a set of lines representing an identical house corner edge. ....	33
Figure 3.12	The house corner center of an house corner image. ....	34
Figure 3.13	The different color points represent the different edge respectively. ....	35
Figure 3.14	Refitting of two non-vertical corner edges.....	37
Figure 3.15	An experimental result of a different house corner. (a) The sets of lines represent an identical house corner edge. (b) The house corner center. (c) The edge points (d) Refitting of two non-vertical corner edges .....	38

Figure 3.16 One more experimental result of another different house corner. (a) The sets of lines represent an identical house corner edge. (b) The house corner center. (c) The edge points (d) Refitting of two non-vertical corner edges	39
Figure 4.1 Flowchart of obstacle avoidance process. ....	41
Figure 4.2 Two coordinate systems in this study. (a) The vehicle coordinate system. (b) The global coordinate system. ....	42
Figure 4.3 Definition of the direction angle (a) $\omega$ is positive. (b) $\omega$ is negative. ....	43
Figure 4.4 The coordinate transformation between the vehicle coordinate system and vehicle coordinate system. ....	44
Figure 4.5 Flowchart of obstacle detection process. ....	45
Figure 4.6 Region growing to find the floor area in an image. (a) The floor image. (b) Detection of the floor area. ....	47
Figure 4.7 Detection of an obstacle within the warning area. (a) An image including an obstacle. (b) The detection of the obstacle. ....	49
Figure 4.8 An illustrated of attaching the lines on the floor. ....	50
Figure 4.9 Using an obstacle image to find the front of the obstacle in the WCS. (a) An obstacle in image coordinate system. (b) The front of the obstacle in world coordinate system. ....	51
Figure 4.10 An illustration of computation of goal-direction minimum path. ....	52
Figure 4.11 Top view of an obstacle avoidance experiment. ....	54
Figure 5.1 RGB 24-bit color cube. (a) Cube with black corner. (c) Cube with white corner [13]. ....	57
Figure 5.2 The HSI color model based on circular color planes. The circles are perpendicular to the vertical intensity axis [13]. ....	58
Figure 5.3 Flowchart of fire detection. ....	61
Figure 5.4 (a) An image without fire. (b) The hue of (a) within the ranges $H_{f1}$ or $H_{f2}$ . (c) The saturation of (a) over than $S_f$ . (d) The intensity of (a) equals $I_f$ . (e) The intersection of (b), (c), (d), and (e). ....	62
Figure 5.5 (a) An image with fire. (b) The hue of (a) within the ranges $H_{f1}$ or $H_{f2}$ . (c) The saturation of (a) over than $S_f$ . (d) The intensity of (a) equals $I_f$ . (e) The intersection of (b), (c), (d), and (e). ....	63
Figure 5.6 The two consecutive images in ordinary situation. ....	65
Figure 5.7 The two consecutive images when the lighting fails. (a) The ordinary image. (b) The image of lighting failure. ....	65
Figure 5.8 The continuous images. (a) The initial situation. (b) The ordinary situation after (a). (c) The image of the lighting failure. (d) The continuous image after (c). ....	67

Figure 5.9 The continuous images. (a) The initial situation. (b) The ordinary situation after (a). (c) The image of the lighting failure. (d) The continuous image after (c).....	68
Figure 5.10 The intensity histograms respective to Figure 5.X. (a) The mean of the intensity is 0.73. (b) The mean of the intensity is 0.73. (c) The mean of the intensity is 0.55. (d) The mean of the intensity is 0.32.....	68
Figure 5.11 Flowchart of lighting failure detection. ....	69
Figure 6.1 A skewed image.....	72
Figure 6.2 A painting on a wall image. ....	76
Figure 6.3 The flowchart of computation of the matrix $H$ .....	79
Figure 6.4 The flowchart of the learning painting process. ....	80
Figure 6.5 The experimental result of learning a painting. (a) A painting on a wall image. (b) A rectangle including the painting. (c) The four borders of the painting. (d) The four intersections computed from the four borders. (e) The four point correspondences. (f) The rectified painting. ....	81
Figure 6.6 The flowchart of the painting matching process. ....	84
Figure 6.7 The experimental results of the painting matching process. (a) and (b) are the two different rectified painting image of the same painting. (c) and (d) are the normalized image in gray-level of (a) and (b) respectively. (e) and (f) are the block image of (c) and (d) respectively. (g) is the overlapping result of (e) and (f). ....	85
Figure 6.8 The experimental results of the painting matching process of two different paintings. (a) and (b) are the two different rectified painting image of the two different paintings. (c) and (d) are the normalized image in gray-level of (a) and (b) respectively. (e) and (f) are the block image of (c) and (d) respectively. (g) is the overlapping result of (e) and (f).....	86
Figure 6.9 The flowchart of the security monitoring process.....	88
Figure 7.1 The used corner and learned image. ....	89
Figure 7.2 An illustration of learned data and navigation path.....	90
Figure 7.3 A navigation process. (a) and (b) are ordinary situations. (c) and (d) are obstacle avoidance. (e) and (f) show the vehicle monitoring a painting. ..	92
Figure 7.4 The experimental result of obstacle avoidance and detection of lighting failure and fire.....	92

# Chapter 1

## Introduction

### 1.1 Motivation

Researches on autonomous vehicles or mobile robots have been conducted for a long time, and there are more and more applications in this domain. Applications of vision-based vehicles are especially useful in human environments, such as nursing robots, entertainment robots, home service robots, security patrolling robots, etc.

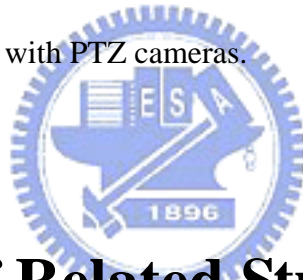
For security surveillance, installation of stationary cameras is a usual method to monitor indoor environments. However, there may exist some areas the camera view cannot cover because the cameras are fixed. This is an obvious disadvantage of conventional security surveillance systems. Fortunately, we can employ a vision-based autonomous vehicle to overcome this weakness because the vehicle is movable. It can navigate around indoor environments to help monitor the areas the camera view cannot cover. Moreover, the use of vehicles has other advantages. For example, it can substitute manpower because a vehicle only spends electronic power or fuel, is able to patrol the whole day, and need not rest like human beings.

In most applications, we must take the advantage of learning to teach a vehicle where to navigate, what to monitor, and how to judge whether an object exists or not, before it starts security patrolling in indoor environments. After the learning process, the vehicle can navigate around the environment and monitor objects autonomously.

While the vehicle navigates routinely, it may encounter some unexpected situations, such as an obstacle or fire in the environment. So a vision-based

autonomous vehicle with capabilities of obstacle avoidance and danger condition detection is necessary in many applications. One of the research goals in this study is to develop a method to detect danger conditions, such as obstacles or fire, and make the vehicle possess the ability to avoid obstacles or to warn people of danger conditions.

With the improvement of the camera technology, a vehicle can be equipped with a pan-tilt-zoom camera (abbreviated as PTZ camera in the sequel) nowadays instead of with a fixed pinhole camera used in the past. Such vision-based autonomous vehicles can help human beings to complete more tasks. By means of the PTZ camera, the view of a vehicle is no longer restricted to be a lower area; instead, it is extended to a wider range. Thus we also propose an approach to monitoring paintings on walls using features in images taken with PTZ cameras.



## 1.2 Survey of Related Studies

In order to let a vehicle navigate automatically and complete the mission of security patrolling in indoor environments, the design of a learning strategy is required. The aim of this step is to learn navigation paths and to record the features of monitored objects. Furthermore, it is very important to know the distance between the vehicle and an object no matter in learning or navigation. A 2D-to-3D distance transformations proposed by Lai and Tsai [1] is to use a curve fitting technique and a modified interpolation technique. Then the distances between the vehicle and the surroundings in real world can be known through captured images. Li and Tsai [2] proposed a graph-based navigation method. The creation of a navigation map is also accomplished through a learning process. Then the vehicle navigates according to the

nodes in the map. The learning process adopted certain complex learning rules to help construct the map. Moreover, Chen and Tsai [3] proposed a fuzzy guidance technique in two navigation modes. It needs to take two kinds of learned data to create the navigation map. After creating the map, a fuzzy guidance technique is applied to accomplish the navigation work. Liu and Tsai [4] proposed a security patrolling navigation method which utilizes multiple-camera vision and autonomous vehicle navigation techniques to patrol in building corridors. Chen and Tsai [5] proposed a learning method by manual driving to teach the vehicle where to navigate and what to monitor. By this method, the vehicle has the ability to navigate in indoor environments and to monitor objects automatically after learning.

The vehicle location is very useful information in navigation, so we need a method to estimate the vehicle location. Installing an odometer on the vehicle is a good idea, but using a vision-based vehicle location estimation method is a better way to tolerate mechanic errors. Fukui [6] proposed a method to locate vehicle locations by placing diamond shapes whose boundary consists of four identical thick line segments with known lengths. Magee and Aggarwal [7] used a sphere on which two perpendicular great circles are drawn as a standard landmark for vehicle location. Chou and Tsai [8] proposed a method which utilized a house corner existing naturally in the house to estimate the vehicle location.

## **1.3 Overview of Proposed Approach**

The aim of this study is to design a vision-based vehicle system for security surveillance in indoor environments with danger condition monitoring ability. An overall framework of the proposed system is illustrated in Figure 1.1. With this

purpose, the system needs not only an ability of navigation but also an ability to pay attention to surrounding environments. Furthermore, design of a simple and flexible learning method is also necessary. In order to achieve these two goals, we use two tools. One is an odometer in the vehicle and the other is the image captured by the camera equipped on the vehicle. The odometer provides the position of the vehicle, and the image helps recognition of objects and monitoring of surrounding environments. We also utilize images to detect house corner for the vehicle location estimation. The vehicle equipped a PTZ camera has the ability to monitor objects at a higher view, such as paintings on walls.

Since the vehicle is a kind of machine, records of the odometer in the vehicle may include mechanic errors after the vehicle navigates a sufficient distance. Calibration of the vehicle location is necessary to prevent the error from accumulating too seriously. We propose a vision-based vehicle location estimation method to adjust the position and direction of the vehicle. This method utilizes the image of a house corner.

The vehicle may also be designed to have the capability of observing surroundings all the time to detect danger conditions immediately. In order to detect danger conditions as soon as possible, a simpler and faster method is necessary. The HSI color of an image of the surrounding provides very good attributes to help the vehicle judge whether the surrounding condition is dangerous or not in real time. We use the camera to capture the surrounding image sequentially and analyze the conditions continuously.

The camera on the vehicle is normally equipped at a lower position. Therefore, when the camera is used to take an image of a higher view, the taken image is always skewed. In order to increase the accuracy of monitoring paintings on walls, we apply an image rectification technique to correct the image skewing. The shape and the ratio

features of a rectified image are more proper for use in security monitoring of paintings. It is necessary to carry out image rectification automatically when the vehicle performs security surveillance. We use the features of paintings on walls and some image processing techniques to obtain rectified images of paintings.

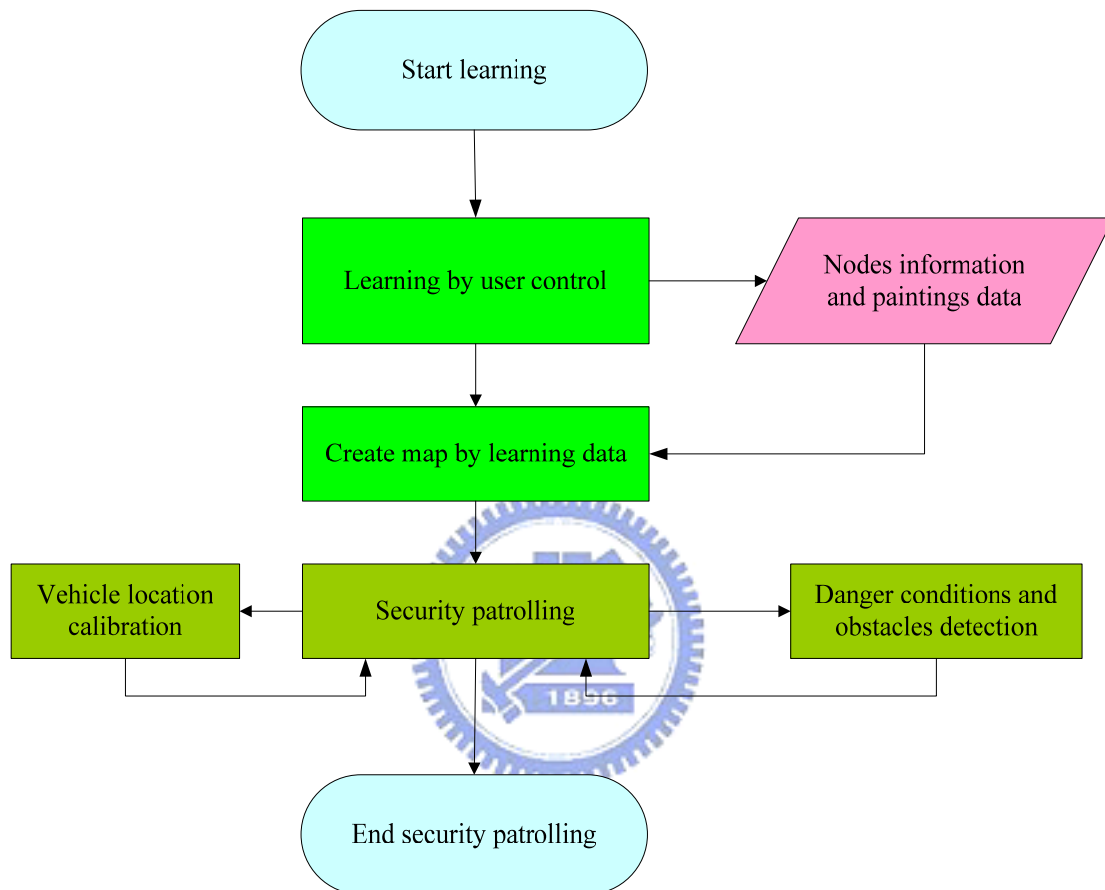


Figure 1.1 Flowchart of proposed system.

## 1.4 Contributions

The main contributions of this study are summarized in the following.


- (1) A vision-based vehicle location estimation method utilizing house corner images to prevent mechanic errors from accumulation is proposed.
- (2) A method to find automatically house corners in images captured by a PTZ



camera is proposed.

- (3) A faster and simpler and less method for real-time obstacle avoidance by goal-directed minimum path following is proposed.
- (4) A method for detecting dangerous fire and lighting failure conditions during security surveillance navigation is proposed.
- (5) An image rectification technique for paintings on walls is proposed.
- (6) A method for security monitoring of paintings on walls is proposed.
- (7) A method for detecting and recognizing paintings on walls automatically using the Hough line transform is proposed.

## 1.5 Thesis Organization



The remainder of this thesis is organized as follows. In Chapter 2, we describe the system configuration of the vehicle used as a test bed, as well as the principle of vehicle guidance with operations of obstacle avoidance and danger condition detection. The proposed method of vision-based vehicle location estimation for correcting records of the odometer in the vehicle is described in Chapter 3. The proposed method of real-time obstacle avoidance by goal-direction minimum path following is described in Chapter 4. In Chapter 5, we discuss two kinds of danger conditions and describe the proposed methods for danger condition monitoring. In Chapter 6, the proposed method for security monitoring of paintings on walls are described. The experimental results are shown in Chapter 7. Finally, some conclusions and suggestions for future works are given in Chapter 8.

# Chapter 2

## System Configuration and Navigation Principles

### 2.1 Overview

Security surveillance in indoor environments is paid more and more attention recently even in ordinary families. Using a vision-based autonomous vehicle to perform security surveillance is a good idea because it can save manpower and the vehicle can patrol all day without taking a rest. In general, there are usually obstacles, narrow paths, and unexpected danger conditions in indoor environments. Therefore, a small and dexterous vehicle is a wonderful choice for this study. A small vehicle is convenient to monitor the space under tables, beds, cabinets, etc., and can navigate for a larger range.

It is indispensable to give the vehicle vision ability to accomplish security surveillance. In this study, we take a pan-tilt-zoom camera as an eye of the vehicle. Although the vision of the vehicle is monocular cause of only one camera, we use computer vision techniques to overcome the problem in this study. In Section 2.2, the features of a pan-tilt-zoom camera and the reason of using such a camera are described in detail.

The vehicle system used in this study as a test bed is composed of a small vehicle and a pan-tilt-zoom camera. There are some communication and control equipments for users to communicate with the vehicle and to control the vehicle to achieve

functions. The entire architecture including hardware and software is introduced in Section 2.3.

We also need a principle to guide the vehicle when the vehicle detects danger conditions in security patrolling in indoor environments. The principle of the vehicle guidance adopted in the study is described in Section 2.4.

## **2.2 Advantage of Using Pan-tilt-zoom IP Cameras**

A traditional pinhole camera used on the vehicle is lack of dexterity because the view of the camera is fixed. It means that if we want to look at another situation out of the view, we should move the vehicle to face the situation first. With a new pan-tilt-zoom camera, we can look at anywhere by turning the camera around without moving the vehicle first. A pan-tilt-zoom camera can perform panning, tilting, and zooming actions as illustrated in Figure 2.1. It means that we can observe surroundings and get the conditions in them quickly without changing the vehicle direction. We also obtain a wider range of field of view because the camera can pan and tilt. Equipped with a pan-tilt-zoom camera on the vehicle overcomes the disadvantage that the view of a vehicle is fixed and restricted to be a lower area. Such a disadvantage is caused by attaching a fixed camera on a small vehicle at a lower position. On the contrary, a vehicle equipped with a PTZ camera can monitor not only objects at lower positions but also objects at higher areas, like paintings.

Furthermore, because we transmit image data in a wireless way, another disadvantage of a traditional pinhole camera is image data transmission through

analog radio, which usually produces noise, especially in a longer transmission. In addition, the image data we deal with is in digital form and not in analog form, so the image data need to be converted into digital form for a computer to deal with. In this study, we adopt an IP camera which transmits image data through a wired network. The image data transmitted by the IP camera is in digital form and with less noise.

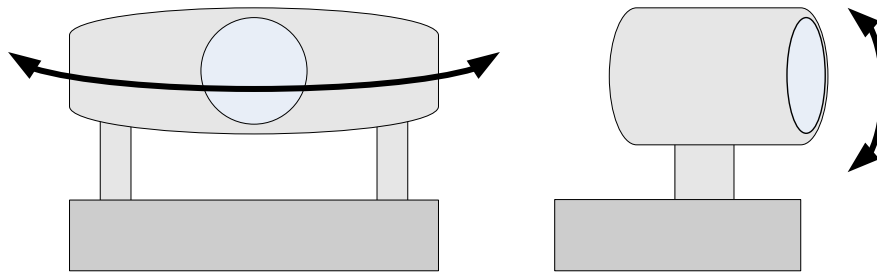


Figure 2.1 Action of a pan-tilt-zoom camera.

## 2.3 System Configuration

In order to perform functions and complete missions in this study, we use a mini-vehicle as a test bed and install a PTZ camera on it as its monocular vision. Because we want to control the whole system remotely, some wireless communication equipments are necessary. We also need a power source to energize the whole system. The hardware architecture and used components of the test bed are described in Section 2.3.1. Besides the hardware, the software is useful to help us develop the system and provide an interface for users to control the vehicle. The software we used in the study is described in Section 2.3.2.

### 2.3.1 Hardware Configuration

Because we control the system remotely, we need an equipment to connect the

camera, the vehicle, and the computer. In this study, the camera communicates through a wired network, the vehicle communicates through a wireless network, and the computer communicates through both. So we use a wireless access point to connect these hardware components. The entire system is illustrated in Figure 2.2 and the appearance of the hardware architecture in our system is shown in Figure 2.3.

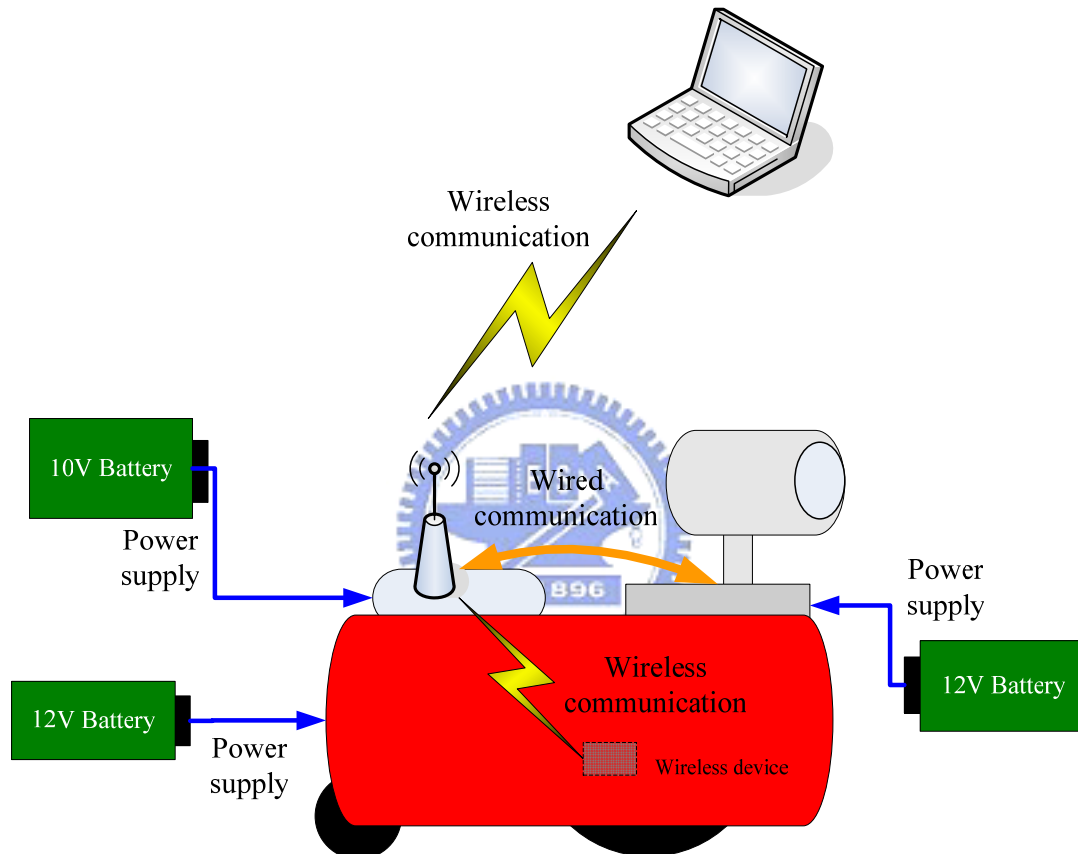


Figure 2.2 Structure of the system

The PTZ IP camera used in this study is AXIS 213 PTZ made by AXIS, as shown in Figure 2.4. This is a camera with a height of 130mm, a width of 104 mm, a depth of 130mm, and a weight of 700g with panning, tilting, and zooming functions. The pan angle range is 340 degrees and the tilt angle range is 100 degrees. It can zoom 26 times optically and 12 times digitally. The image grabbed in our experiments is of the resolution of 320×240 pixels. It also provides a 10Base/100BaseTX Ethernet network and needs an external power source of 13V DC and 1.8A. Because the

camera is not directly connected to the computer used to control the system, we use a wireless access point to connect these two components.



(a)



(b)



(c)

Figure 2.3 The vehicle used in this study. (a) The bevel view of the vehicle. (b) The front view of the vehicle. (c) The left side view of the vehicle.

The mini-vehicle used in this study is an Amigo Robot made by ActivMedia Robotics Technologies, Inc. The length, width, and height of the vehicle are 33cm, 28cm, and 21cm, respectively. There are two larger wheels and one auxiliary small wheel at the tail of the vehicle. The maximum speed of the vehicle is 75cm/sec and the maximum rotation speed is 300 degrees/sec. There are eight ultrasonic sensors, an odometer, and an embedded hardware system in this mobile vehicle. The ultrasonic sensors are not used in this study. The odometer provides the coordinates and the

direction of the vehicle in the navigation. The origin of the coordinates is the starting position of the vehicle. There is a 12V battery in the vehicle to supply the power of the vehicle system.



(a)



(b)



(c)

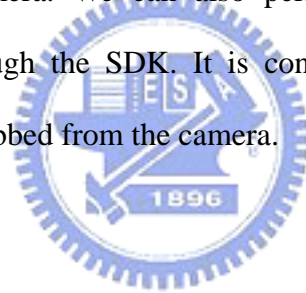
Figure 2.4 The pan-tilt-zoom camera used in this study. (a) The bevel view of the camera. (b) The front view of the camera. (c) The right side view of the camera.

There is one wireless device in the vehicle and another in the PC. The commands of the remote system are transmitted to the wireless signal receiver by an access point that meets the 802.11b standard. By using the access point as a medium, the commands can be transmitted from the computer to the vehicle and the camera.

## 2.3.2 Software Configuration

We use the ARIA, which is an object-oriented programming interface for the ActivMedia Robotics' line of intelligent mobile robots to control the vehicle actions easily, such as move, rotation, and stop. The lowest-level data and information of the vehicle are also retrieved easily by means of the ARIA interface. In other words, any developer can use the ARIA as an interface to communicate with the embedded system of the vehicle. And we use the Borland C<sup>++</sup> builder as the development tool in our experiments.

The AXIS Company also provides a development tool called AXIS Media Control SDK for AXIS 213 PTZ. With the SDK, we can get image previews and the current image from the camera. We can also perform the panning, tilting, and zooming actions easily through the SDK. It is convenient for us to develop any functions with the images grabbed from the camera.



## 2.4 Vehicle Guidance Principle

Before the vehicle patrols in an indoor environment, a navigation map must be created first. The node information of a learned path is indispensable for creating the navigation map. There are two types of nodes in this study. One is used for correction of the vehicle location. The other is used for security monitoring of paintings. We save the node information and camera position when the vehicle moves to a node for correction of the vehicle location. We also save the node information, camera position and painting data when the vehicle moves to a node of security monitoring of a painting. After a manual learning process, a set of learned data, including node



information, are saved in the system. The navigation map is created by using the node information and is used to guide the vehicle in indoor environments. In order to guide the vehicle along the learned path, the vehicle moves sequentially from one node to another according to the navigation map. While the vehicle patrols in indoor environments, there are three things which require attention. One is whether the vehicle reaches the next node or not. The next is whether the vehicle encounters an obstacle or not. The last is whether the vehicle detects any danger conditions or not. An illustration of the vehicle navigation process is shown in Figure 2.5.

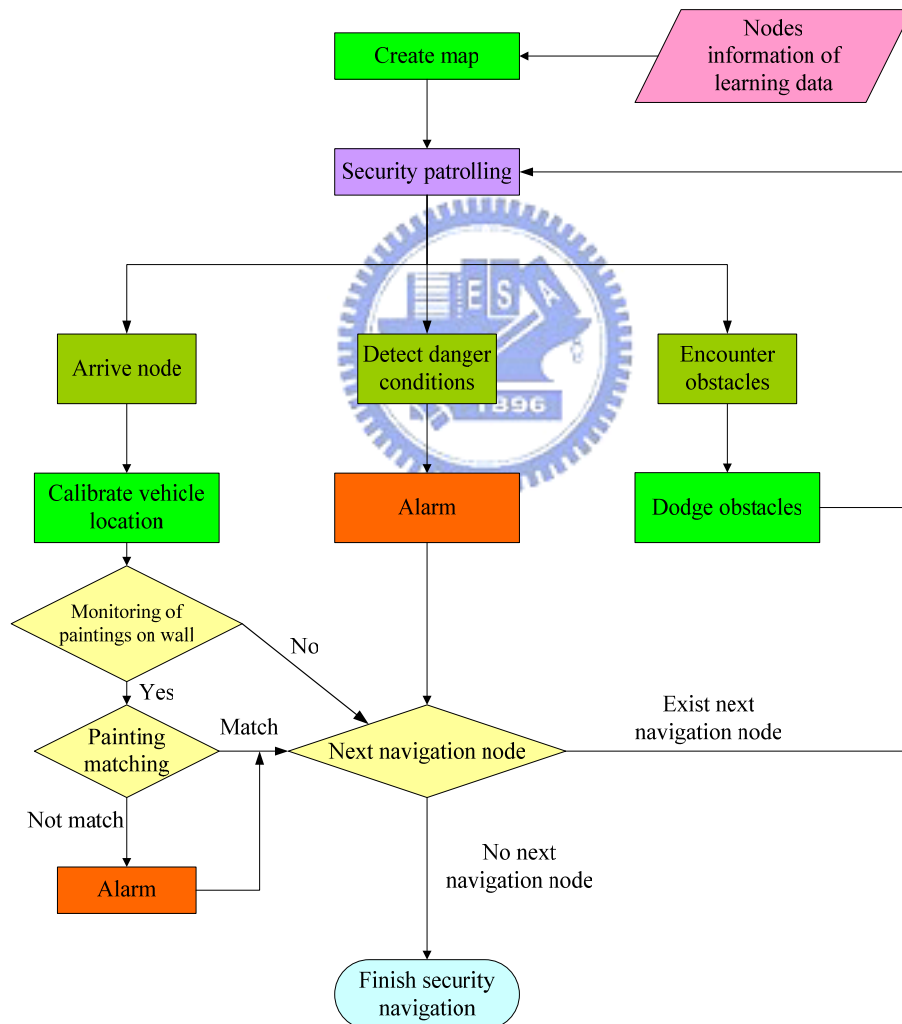


Figure 2.5 Flowchart of proposed navigation process

When the vehicle reaches the next node, it corrects its location by a house corner first. If there is a nearby painting which is monitored, the vehicle uses learned

painting data to detect whether the painting still exists or not. If the detection process of the painting fails, the system issues a warning signal to the user.

The vehicle might encounter a variety of situations in patrolling. In order to react to conditions in real time, we analyze the images grabbed by the camera continuously to detect the following two conditions. If an obstacle is detected on the patrolling path, a new path for obstacle avoidance and destination approaching is planned. If a danger condition is detected, a warning message is announced to relevant people who can prevent the danger condition from getting worse. The proposed vehicle guidance method is described in the following as an algorithm.

**Algorithm 2.1.** *Vehicle guidance procedure.*

*Input:* A set of learned navigation nodes.

*Output:* The vehicle moving action.

*Steps:*

- Step 1. Create the navigation map by the learned navigation nodes.
- Step 2. Guide the vehicle to the next navigation node and detect obstacles or danger conditions such as fire and lighting failure.
- Step 3. Whenever the vehicle reaches a navigation node, move the camera to face a house corner.
- Step 4. Estimate the vehicle location by the house corner and correct the odometer of the vehicle.
- Step 5. If the location of the vehicle indicates that the vehicle is too far from the navigation node, then guide the vehicle to the navigation node.
- Step 6. Go to step 2 if there is a next navigation node.

# Chapter 3

## Vehicle Guidance by Vehicle Location Estimation Using A House Corner Detection Technique

### 3.1 Overview

The location of a vehicle is important information to guide the vehicle to navigate correctly in an indoor environment. Installation of an odometer on a vehicle is useful to record the position and the orientation of the vehicle in every navigation cycle. Because of the existence of possible mechanic errors, the position data obtained from the odometer might be inconsistent with the real position. In order to make a vehicle navigate more stably, vision-based vehicle location estimation is helpful to correct the error.

Using binocular cameras to simulate human stereo vision for vehicle location estimation is a method. But in a controllable indoor environment, using a monocular camera is sufficient for vehicle location estimation if special standard marks are adopted. This kind of technique depends on the use of image analysis under certain reasonable assumptions. A number of methods will be reviewed in Section 3.2. One of these methods is vehicle location estimation by house corner detection proposed by Chou and Tsai [8]. The proposed vehicle location estimation method in this study is a simplified version derived from Chou and Tsai. The principle and feasibility of the method are described in Section 3.3.

In order to detect house corner edges and to determine automatically the coefficients of the edge line equations which are used in the vehicle location estimation, several image processing techniques are employed. The techniques and processes are described in Section 3.4.

## **3.2 Review of Vehicle Location Estimation Techniques**

In a known indoor environment, the use of special landmarks and monocular images captured by a camera is an ordinary technique for vehicle location estimation. As shown in Figure 3.1 (a), a diamond shape whose boundary consists of four identical thick line segments all with a known length is the standard landmark proposed by Fukui [6]. The two diagonals are arranged to be vertical and horizontal, respectively. However, one restriction is that the lens center of a camera should be put at the same height as the diamond center and the camera optical axis is pointed through the diamond center. Under the restriction, the boundary of the diamond images taken by the camera is extracted, and the lengths of the two diagonals are computed. Based on the two diagonals in the image, the location of the camera is finally derived, i.e., the location of vehicle is obtained. Courtney and Aggarwal [9-10] relaxed the restriction of the method by a reasonable assumption that the height of the camera is known.

Another method proposed by Magee and Aggarwal [7] is to take a sphere on which two perpendicular great circles are drawn as a standard landmark for vehicle location estimation. An illustrated is in Figure 3.1 (b). This method also possesses the

same restriction that the camera optical axis must be pointed through the sphere center before the image of the sphere is taken. According to the size of the projected circle of the sphere, the distance from the camera to the sphere center and the direction of the camera are computed. From the points on the projections of the great circles that are closest to the center of the sphere outline, the vehicle location is finally computed accordingly in terms of the sphere coordinates.

Chou and Tsai [8] utilized a set of house corner edges as a standard landmark. This landmark exists naturally in a house, and is not entirely artificial-created. Such a landmark is visible from a house floor, and most show as an identical geometric structure of a “Y” shape like Figure 3.1 (c). The projections of the three lines going through the corner on the image plane are then extracted to estimate the vehicle location. A reasonable assumption is the distance from the camera to the ceiling is known. Under this assumption, the coefficients of the equations of the house corner edge lines are substituted into a set of location formulas. Finally we can estimate the vehicle location by results yielded by the location formula set.

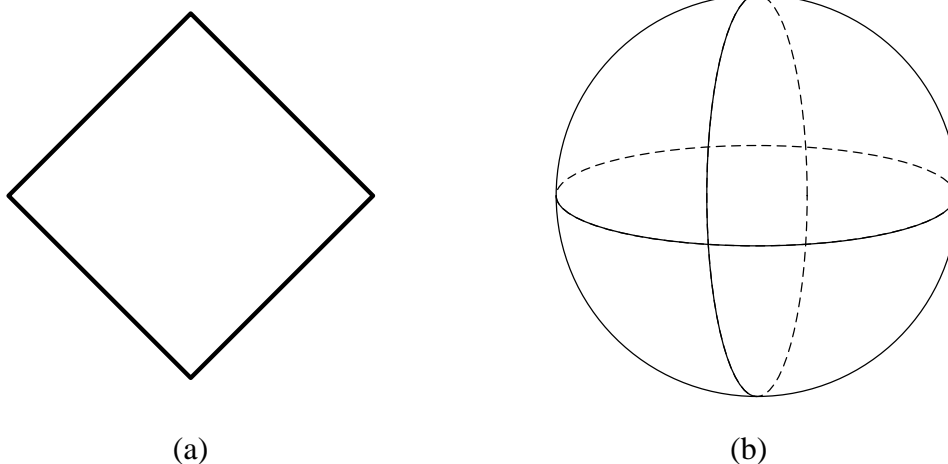


Figure 3.1 Three different landmarks. (a) A diamond-shaped standard mark used in [6]. (b) A sphere used for robot location in [7]. (c) The perspective projection of a house corner used in Chou and Tsai [8].

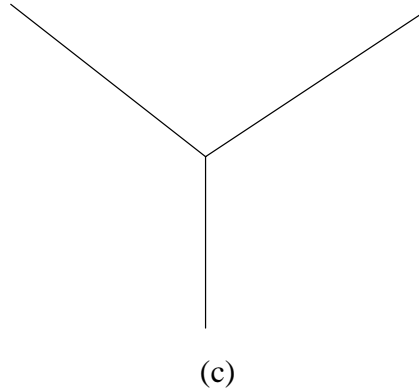


Figure 3.1 Three different landmarks. (a) A diamond-shaped standard mark used in [6]. (b) A sphere used for robot location in [7]. (c) The perspective projection of a house corner used in Chou and Tsai [8].

### 3.3 Vehicle Location Estimation by House Corner Detection

We use only two edge lines instead of all the three ones of a house corner to estimate the vehicle location. The pattern used in the study is illustrated in Figure 3.2. The vehicle is equipped with a PTZ camera to get a house corner image in each navigation cycle. In order to describe the proposed method conveniently, we introduce some coordinate systems in Section 3.3.1. The principle is to use the coefficients of the equations of the edge lines through the corner point to estimate the vehicle location. The detail of the principle is described in Section 3.3.2.

The relation between the vehicle location parameters and the coefficients of the edge line equations are discussed in detail in Section 3.3.3. The derivation of the vehicle location parameters is also described.

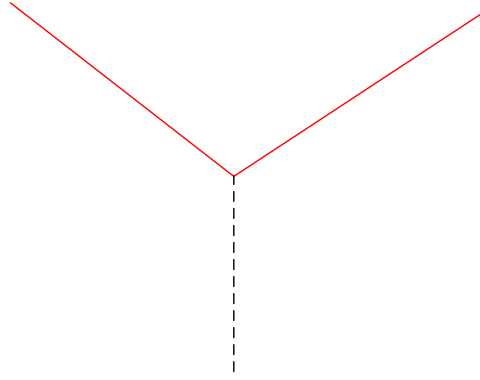


Figure 3.2 The two red lines of the perspective projection of a house corner used in the study.

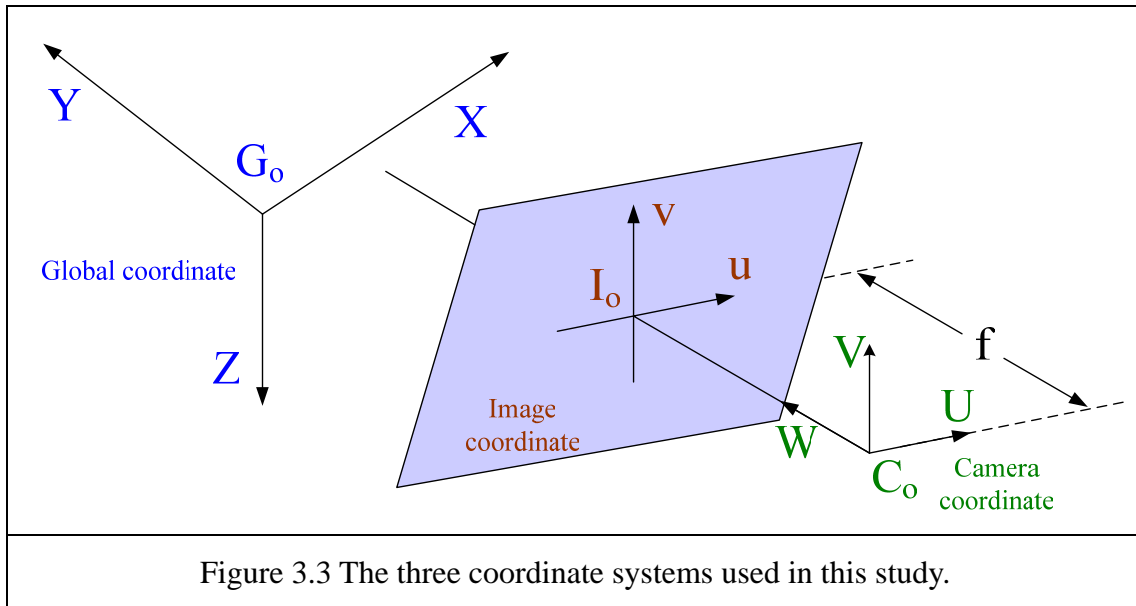
### 3.3.1 Coordinate Systems

The three coordinate systems used in this study are introduced at first. By these coordinate systems, it will be clear and convenient to describe a vehicle location. The definitions of the three coordinate systems are described in the following.

- (1) The global coordinate system (GCS, denoted as  $X$ - $Y$ - $Z$ ): In the global coordinate system, we assume that the two perpendicular lines of a corner on the ceiling are the  $X$  and  $Y$  axes, the vertical line is the  $Z$  axis, and the corner point is the origin  $G_o$  of the global coordinate system.
- (2) The camera coordinate system (CCS, denoted as  $U$ - $V$ - $W$ ): In the camera coordinate system, we establish the  $U$ ,  $V$ , and  $W$  axes. The  $U$ - $V$  plane is parallel to the image plane, and the  $U$  axis is parallel to the  $X$ - $Y$  plane of the global coordinate system. The origin  $C_o$  is located at the camera lens center and the  $W$  axis is aligned to be parallel with the camera optical axis.
- (3) The image coordinate system (ICS, denoted as  $u$ - $v$ ): The image plane is located at  $W = f$ , where  $f$  is the focus length of the camera. The image plane of the image coordinate system is coincident with the  $u$ - $v$  plane and the

origin  $I_o$  is the image plane center.

The relations among the three coordinate systems are illustrated in Figure 3.3.



### 3.3.2 Principle of Proposed Method

The three equations of the edge lines through the corner point in terms of image coordinates  $(u, v)$  are described by  $u_p + b_i v_p + c_i = 0$ , where  $i = 1, 2, 3$ . The desired vehicle location will be described by three position parameters  $X_c, Y_c$ , and  $Z_c$  and two direction parameters  $\psi$  and  $\theta$ , where  $Z_c$  is the distance from the camera to the ceiling and is assumed to be known,  $\theta$  is the pan angle of a camera, and  $\psi$  is the tilt angle of the camera with respect to the global coordinate system. The five vehicle location parameters can be derived in terms of the four coefficients  $b_1, c_1, b_2$  and  $c_2$  of the edge line equations in the corner image. Finally the vehicle location could be estimated after the coefficients of the edge line equations are computed.



### 3.3.3 Location Estimation by Coefficients of Edge Line Equations

In this section, we will derive the relation between the global coordinates and the coefficients of the edge line equations in the image coordinate system. At first, we transform the global coordinates into the camera coordinates. The transformation consists of four steps.

Step 1 Translate the origin of the global system by  $(X_c, Y_c, Z_c)$  to the origin of the camera system in the following way:

$$T(X_c, Y_c, Z_c) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -X_c & -Y_c & -Z_c & 1 \end{bmatrix}. \quad (3.1)$$

Step 2 Rotate the  $X-Y$  plane about the  $Z$  axis through the pan angle  $\psi$  in the following way such that the  $X-Y$  plane is parallel to the  $U-V$  plane:

$$R_z(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 & 0 \\ \sin \psi & \cos \psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.2)$$

Step 3 Rotate the  $Y-Z$  plane about the  $X$  axis through the tilt angle  $\theta$  in the following way such that the  $X-Y$  plane is parallel to the  $U-V$  plane:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.3)$$

Step 4 Reverse the  $Z$  axis in the following way such that the positive direction of the  $Z$  axis is identical to the negative direction of the  $W$  axis

$$F_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.4)$$

Now, if we have a point  $P$  with global coordinates  $(x, y, z)$  and camera coordinates  $(u, v, w)$ , the transformation in the two coordinate systems can be described in short as:

$$(u, v, w, 1) = (x, y, z, 1) \cdot T(X_c, Y_c, Z_c) \cdot R_z(\psi) \cdot R_x(\theta) \cdot F_z = (x, y, z, 1) T_r \quad (3.5)$$

where

$$T_r = T(X_c, Y_c, Z_c) \cdot R_z(\psi) \cdot R_x(\theta) \cdot F_z$$

$$= \begin{bmatrix} \cos \psi & -\sin \psi \cos \theta & -\sin \psi \sin \theta & 0 \\ \sin \psi & \cos \psi \cos \theta & \cos \psi \sin \theta & 0 \\ 0 & \sin \theta & -\cos \theta & 0 \\ x_0 & y_0 & z_0 & 1 \end{bmatrix} \quad (3.6)$$

with

$$x_0 = -X_c \cos \psi - Y_c \sin \psi ;$$

$$y_0 = (X_c \sin \psi - Y_c \cos \psi) \cos \theta - Z_c \sin \theta ; \quad (3.7)$$

$$z_0 = (X_c \sin \psi - Y_c \cos \psi) \sin \theta + Z_c \cos \theta .$$

If  $P$  is on the  $X$  axis with global coordinates  $(x, 0, 0)$ , the camera coordinates  $(u_x, v_x, w_x)$  can be rewritten as:

$$(u_x, v_x, w_x, 1) = (x, 0, 0, 1) \cdot T_r$$

$$= (x \cos \psi + x_0, -x \sin \psi \cos \theta + y_0, -x \sin \psi \sin \theta + z_0, 1). \quad (3.8)$$

Let  $(u_p, v_p)$  be the image coordinates of the projection of  $P$ , then we have the following two equations to describe  $u_p$  and  $v_p$  according to the triangulation principle:

$$u_p = \frac{f \cdot u_x}{w_x}; \quad (3.9)$$

$$v_p = \frac{f \cdot v_x}{w_x}, \quad (3.10)$$

where  $f$  is the camera focus length.

Substituting the values of  $u_x$  and  $v_x$ , we may rewrite Formulas (3.9) and (3.10) above for  $u_p$  and  $v_p$  as Equations 3.11 and 3.12 below:

$$u_p = \frac{f \cdot (x \cos \psi + x_0)}{-\sin \psi \sin \theta + z_0}; \quad (3.11)$$

$$v_p = \frac{f \cdot (-x \sin \psi \cos \theta + y_0)}{-\sin \psi \sin \theta + z_0}. \quad (3.12)$$

Eliminating the variable  $x$ , we can get the equation for the projection of the X-axis in the image plane as Equation 3.13 below:

$$u_p = \frac{v_p \cdot (-z_0 \cos \psi - x_0 \sin \psi \sin \theta) + f \cdot (y_0 \cos \psi + x_0 \sin \psi \cos \theta)}{-y_0 \sin \psi \sin \theta + z_0 \sin \psi \cos \theta}. \quad (3.13)$$

After substituting Equation 3.13 into  $u_p + b_1 v_p + c_1 = 0$  and substituting Equation 3.7 for  $x_0$ ,  $y_0$ , and  $z_0$ , we obtain the coefficients  $b_1$  and  $c_1$  as described by Equations 3.14 and 3.15 below:

$$b_1 = \frac{Y_c \sin \theta - Z_c \cos \psi \cos \theta}{-Z_c \sin \psi}; \quad (3.14)$$

$$c_1 = \frac{f \cdot (-Y_c \cos \theta - Z_c \cos \psi \sin \theta)}{-Z_c \sin \psi}. \quad (3.15)$$

Similarly, the coefficients of the edge line equation  $u_p + b_2 v_p + c_2 = 0$  for the projection of the Y axis in the image plane can be derived to be as follows:

$$b_2 = \frac{-X_c \sin \theta - Z_c \sin \psi \cos \theta}{Z_c \cos \psi}; \quad (3.16)$$

$$c_2 = \frac{f \cdot (X_c \cos \theta - Z_c \sin \psi \sin \theta)}{Z_c \cos \psi}. \quad (3.17)$$

The next step is to determine the variables of directions,  $\psi$  and  $\theta$ . First, we use Equations 3.14 and 3.15 to derive the variable  $\psi$ . Equations 3.14 and 3.15 can be transformed into Equations 3.18 and 3.19, respectively, described below:

$$Z_c(-b_1 \sin \psi + \cos \psi \cos \theta) = Y_c \sin \theta; \quad (3.18)$$

$$Z_c(-c_1 \sin \psi + f \cos \psi \cos \theta) = -fY_c \cos \theta. \quad (3.19)$$

By eliminating  $Z_c$  and  $Y_c$  from Equations 3.18 and 3.19,  $\tan \psi$  can be described as Equation 3.20 below:

$$\tan \psi = \frac{f}{fb_1 \cos \theta + c_1 \sin \theta}. \quad (3.20)$$

Similarly, Equations 3.16 and 3.17 can be transformed into Equations 3.21 and 3.22, respectively, described below:

$$Z_c(b_2 \cos \psi + \sin \psi \cos \theta) = -X_c \sin \theta; \quad (3.21)$$

$$Z_c(c_2 \cos \psi + f \sin \psi \sin \theta) = fX_c \cos \theta. \quad (3.22)$$

By eliminating  $Z_c$  and  $X_c$  from Equation 3.21 and 3.22,  $\tan \psi$  can be rewritten as Equation 3.23 below shows:

$$\tan \psi = \frac{-fb_2 \cos \theta - c_2 \sin \theta}{f}. \quad (3.23)$$

We can use Equations 3.20 and 3.23 to get  $\tan \theta$  as follows:

$$\tan \theta = \frac{-f(b_1 + b_2)}{c_1 + c_2}. \quad (3.24)$$

Thereafter, we can substitute the value  $\theta$  into Equation 3.23 to obtain the value  $\psi$ . Finally, the values of  $\theta$ ,  $\psi$ , and  $Z_c$  are all known, and can be substituted into Equations 3.14 and 3.16 to obtain the values of  $Y_c$  and  $X_c$ , respectively.

## 3.4 Image Processing for Vehicle Location Estimation

In order to conduct vehicle location estimation by house corner edges automatically, we need a series of image processing and numerical analyses to find the coefficients of the edge line equations in an image. A flowchart of the image processing and numerical analysis process is shown in Figure 3.4.

The main technique is to use the Hough transform to detect lines in an image. The process to find the house corner edges of an image captured from the PTZ camera is described in Section 3.4.1. After we find the house corner edges, the next step is to find the coefficients of the house corner edges. In order to improve the accuracy of the coefficients of the edge line equations, we use a *constrained line refitting* technique to refit the extracted lines into new ones. The constrained condition is that each edge line should go through the house corner point. Finally, since it is sufficient to estimate a vehicle location by only two non-vertical corner edges, we keep the necessary edges from the result of the Hough transform. Then the coefficients of the two non-vertical corner edges are computed. By the coefficients, the vehicle location can be estimated. In Section 3.4.2, we will discuss the proposed constrained line refitting technique and the restrictions for keeping two non-vertical edges.

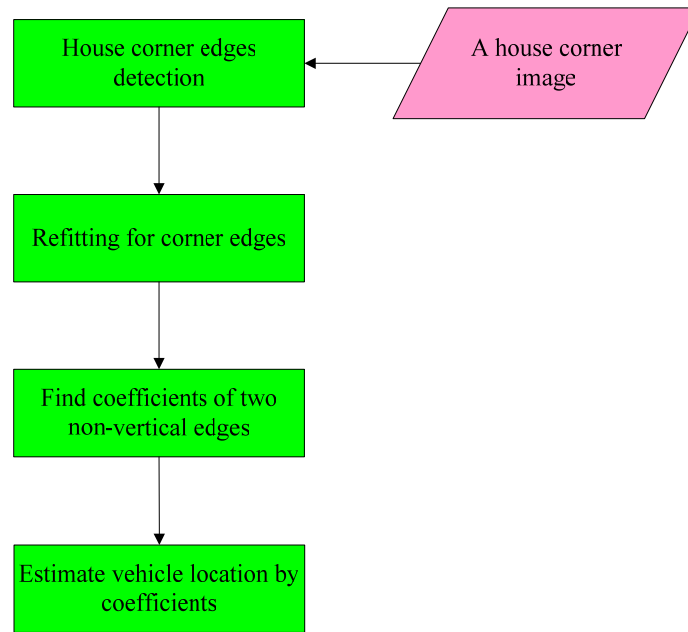


Figure 3.4 Flowchart of vehicle location estimation.

### 3.4.1 Detection of House Corner Edges

In order to detect the edges of a house corner in an acquired image, we apply several image processing techniques to find out the edges. A flowchart is illustrated in Figure 3.5.

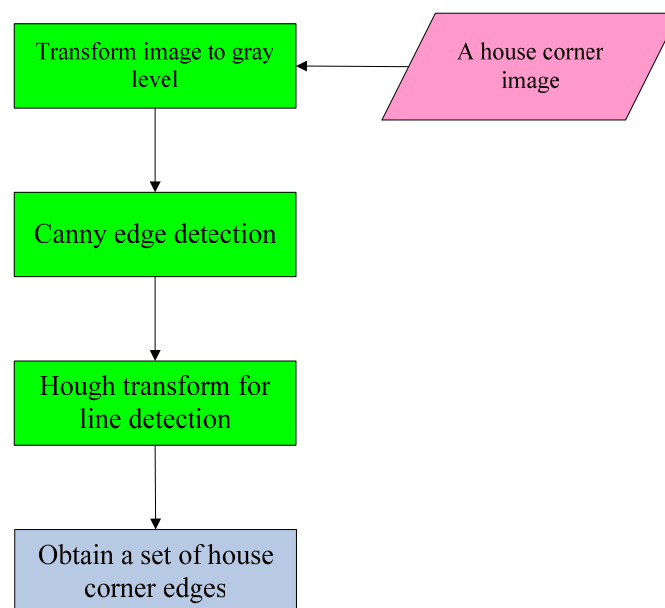


Figure 3.5 Flowchart of house corner edges detection

A house corner image captured by a PTZ camera may contain not only house corner edges but also a lot of other components. But we only need the house corner edges. The Hough transform is a powerful technique to detect straight lines in an image. In the Hough transform, each line is represented in the form of Equation 3.25 below:

$$r = x \cos \alpha + y \sin \alpha \quad (3.25)$$

where  $x$  and  $y$  are the image coordinates of the line points,  $\alpha$  is the direction of the normal of the line, and  $r$  is the distance of the line to the origin of the image coordinate system. Each point in the image coordinate system is transformed into a curve with  $\alpha$  and  $r$  as the coordinates. All the curves corresponding to a set of image points lying on a straight line will intersect at a point. Because the Hough transform is useful to detect straight lines, we use the technique to detect house corner edges in this study.

Before applying the Hough transform, we should find out the edges in an image. The Canny edge detection can help us to detect edges by an appropriate threshold. Figure 3.6 (a) is a house corner images and Figure 3.6 (b) is the result image after the Canny edge detection was applied respectively. Apparently the result shows the edge lines of the house corner clearly. However, it still contains other unconcerned edges.

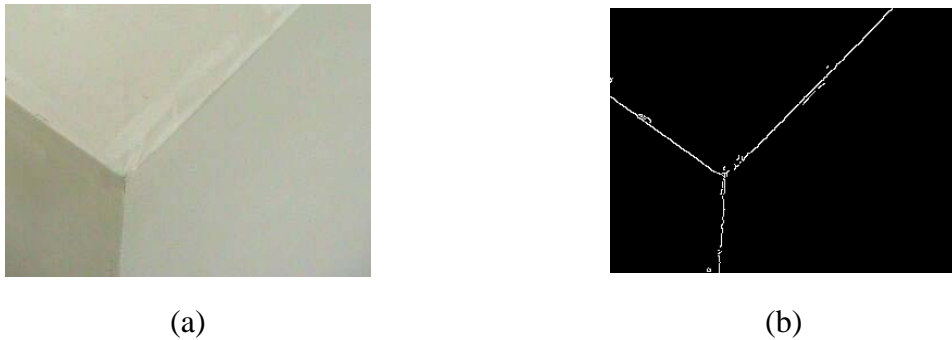


Figure 3.6 (a) The color corner images. (b) The Canny result images of (a).

Because the house corner edges always extend to the boundary from the house corner point in an image, we can apply the Hough transform by a proper threshold, meaning that lines in an image must be long enough, to be regarded as house corner edges. Each line detected by the Hough transform is represented by the two parameters  $\alpha$  and  $\gamma$ . Figure 3.7 shows the house corner edge lines after the Hough transform. Then we can obtain a set of house corner edge data represented by  $\alpha$  and  $\gamma$ . An algorithm including the above details is described in the following.

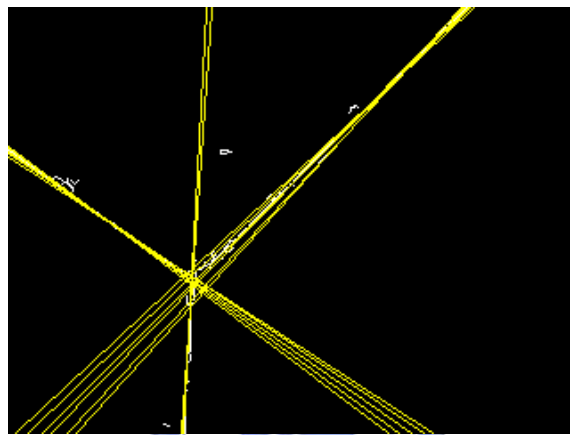


Figure 3.7 The image after applying Hough transform in Figure 3.6 (b).

**Algorithm 3.1.** *House corner edges detection.*

*Input:* A house corner image  $I_{corner}$ .

*Output:* A set of house corner edge data  $E_{corner}$ .

*Steps:*

- Step 1. Convert the image  $I_{corner}$  into a gray-level image  $I_{gray}$ .
- Step 2. Apply the Canny edge detection by the threshold  $Th_{canny}$  on  $I_{gray}$  to obtain a set of edges  $E_{corner}$  in  $I_{gray}$ .
- Step 3. Apply the Hough transform by the threshold  $Th_{hough}$ ; reserve each detected line which is longer than  $Th_{hough}$ ; and represent it by the parameters  $\alpha$  and  $\gamma$ .

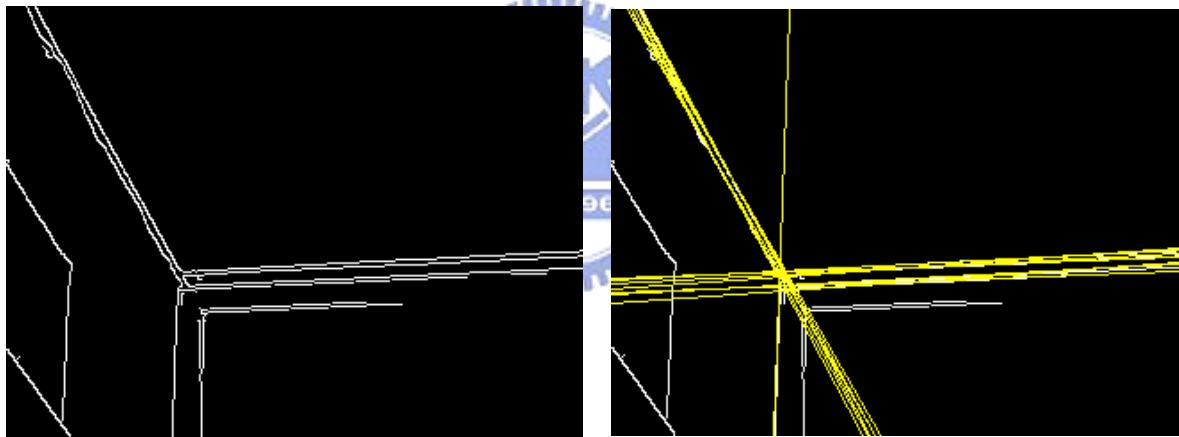


Step 4. Take the final results of house corner edges represented by  $\alpha$  and  $\gamma$  as the desired set  $E_{corner}$ .

More experimental results of corner edge detection shown in Figure 3.8 and Figure 3.9.



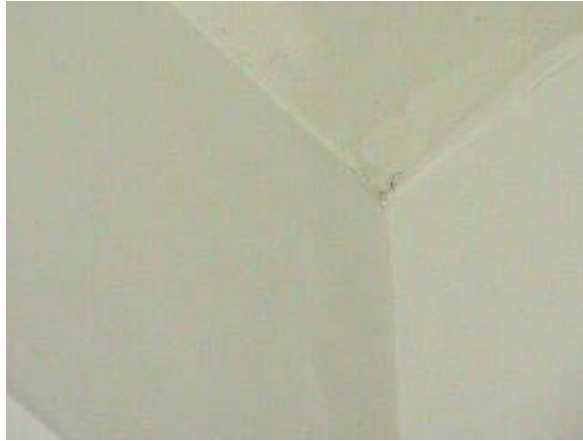
(a)



(b)

(c)

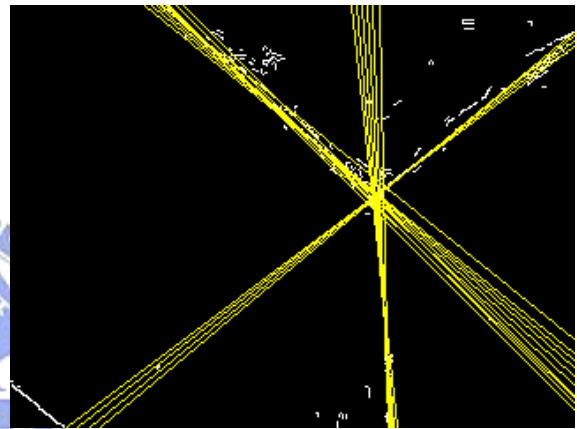
Figure 3.8 An experimental result of a different house corner. (a) Another corner image. (b) The result of applying Canny edge detection of (a). The result of applying Hough transform of (b).



(a)



(b)



(c)

Figure 3.9 One more experimental result of another different house corner. (a) Another corner image. (b) The result of applying Canny edge detection of (a). The result of applying Hough transform of (b).

### **3.4.2 Constrained Line Refitting and Determination of Non-vertical House Corner Edges**

Because the accuracy of vehicle location estimation depends on the precision of the coefficients of edge line equations, we use a constrained line refitting technique to reconstruct the house corner edge lines in an image. This process can improve the

precision of house corner edge lines detected in an image. Moreover, in this study, two non-vertical corner edges are sufficient to estimate the vehicle location, so we have to find out them in the three edges of each corner image. The process for constrained line fitting and determination of non-vertical house corner edges is shown in Figure 3.10.

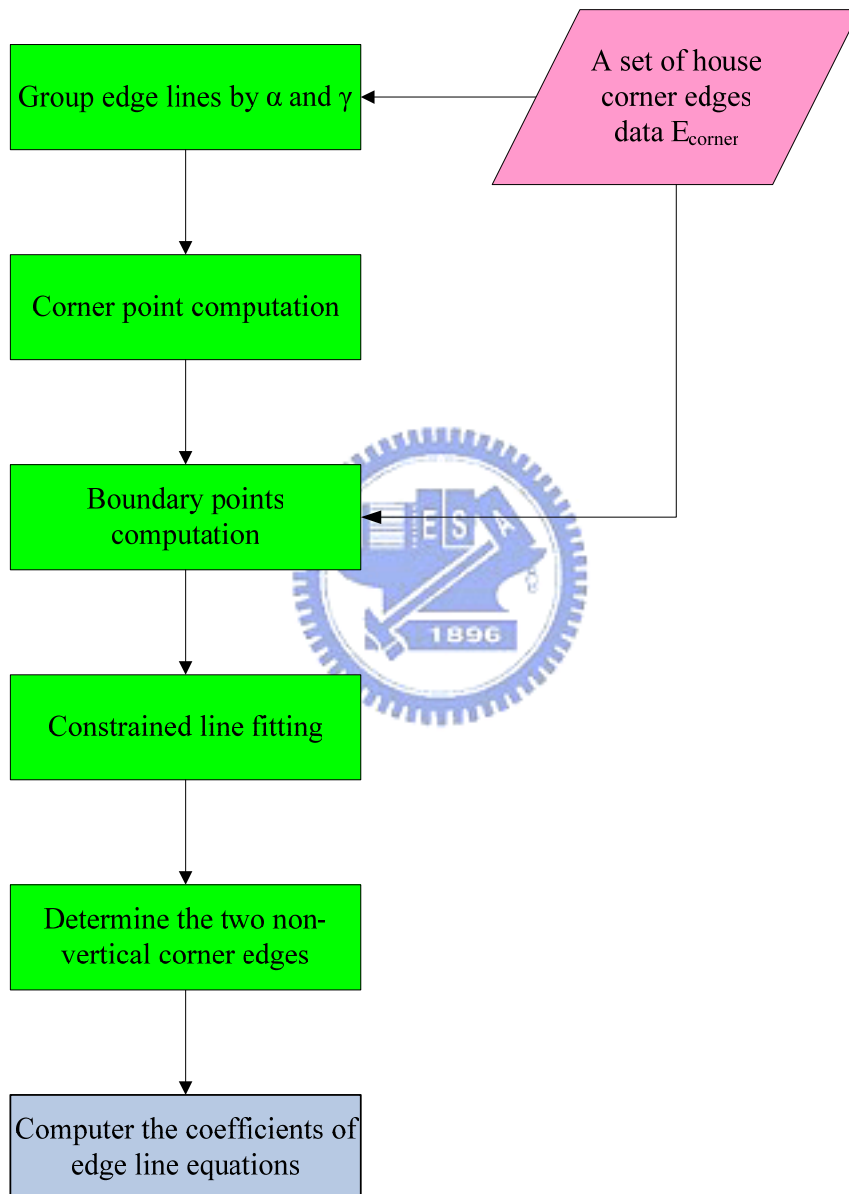


Figure 3.10 Flowchart of the coefficients of corner edge equation computation.

After the Canny edge detection and Hough transform are applied, corner edges are found in a house corner image. However, a found house corner edge may not

represent exactly a line in the image. In normal cases, after applying the Hough transform to a house corner image, more than three corner edge lines are detected. Thus, to group together lines with each group representing an edge line is useful for later processing.

We utilize the features of the Hough transform results, namely, the parameters  $\alpha$  and  $\gamma$  of each detected line. We use these two parameters to group corner edge lines. If the parameter values  $\alpha$  and  $\gamma$  of two lines are closer enough, then they are taken into the same group. Lines in an identical group mean that they represent the same house corner edge actually. An example is shown in Figure 3.11. The red lines come from a corner edge, and the blue line is another corner edge, and the green lines come for the third corner edge.

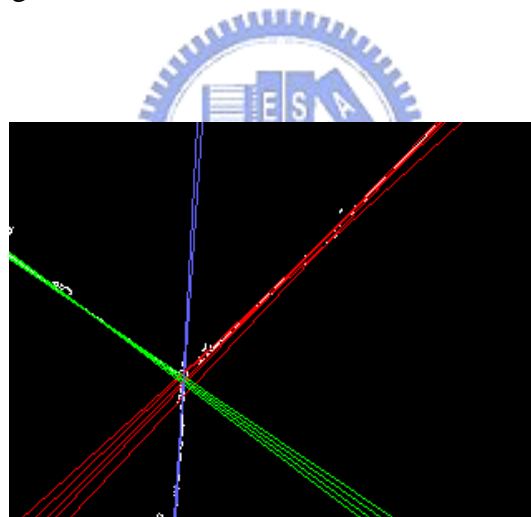


Figure 3.11 Illustration of a set of lines representing an identical house corner edge.

Now, we compute the intersections of all the detected lines, collect their image coordinates, and compute the average image coordinates as a new house corner point. At first, we transform lines described in terms of coordinates  $\alpha$  and  $\gamma$  into ones described in terms of image coordinates. We represent each line as follows:

$$u + bv + c = 0, \quad (3.26)$$

where

$$b = \frac{1}{\tan \alpha}; \quad (3.27)$$

$$c = \frac{\gamma}{\sin \alpha}. \quad (3.28)$$

Then, the intersections of the lines in different groups can be computed by Cramer's rule. Each intersection point of two lines described by  $u + b_1v + c_1 = 0$  and  $u + b_2v + c_2 = 0$  may be described as  $(u, v)$  where  $u = \frac{c_1b_2 - c_2b_1}{b_2 - b_1}$  and  $v = \frac{a_1b_2 - a_2b_1}{b_2 - b_1}$  after some derivations.

We then compute the mean of the intersections in terms of their image coordinates to get  $P_{center} = (u_c, v_c)$  where  $u_c = \frac{\sum_{i=1}^n u_i}{n}$ ,  $v_c = \frac{\sum_{i=1}^n v_i}{n}$  and  $n$  is the total number of intersections. We take this point  $P_{center}$  as a final house corner point.

An example of the house corner point so found is shown in Figure 3.12. The new house corner point is then used to refit house corner edges later.



Figure 3.12 The house corner center of an house corner image.

After we separate the three groups of house corner edges, we can take the pixels in the same group as the points of an edge line. Let the line points to be refitted are denoted by  $S = \{(u_i, v_i), i = 1, 2, \dots, n\}$ . An example of the set is shown in Figure 3.13. Let the refitting line  $L$  be represented as follows:

$$u + b'v + c' = 0. \quad (3.29)$$

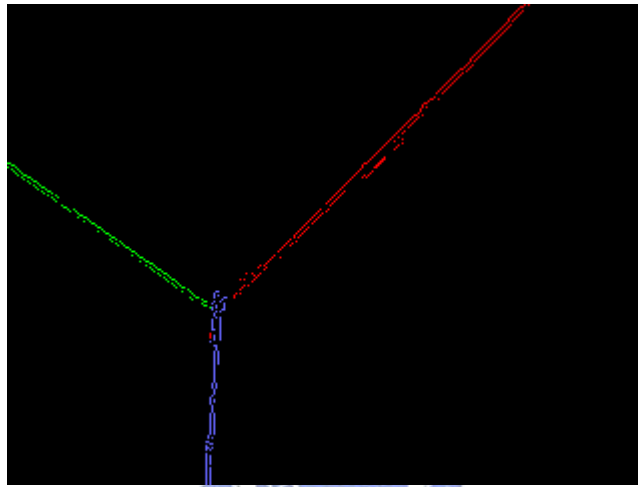


Figure 3.13 The different color points represent the different edge respectively.

By the least-square-error fitting criterion, the line  $L$  satisfies that the sum of the squares of all the distances from the points in  $S$  to  $L$  is minimum and that the parameter  $b'$  satisfies the following formula:

$$b' = \bar{V} - c'\bar{U}, \quad (3.30)$$

where

$$\bar{U} = \frac{\sum_{i=1}^n u_i}{n}; \quad (3.31)$$

$$\bar{V} = \frac{\sum_{i=1}^n v_i}{n}, \quad (3.32)$$

with  $n$  being the number of elements in  $S$ .

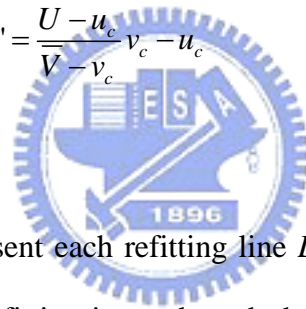
We also constrain the line  $L$  to go through the house corner point  $P_{center}$ . Thus, we can substitute the  $P_{center}$  into Equation 3.29 and describe  $c'$  as follows:

$$c' = -(u_c + b'v_c). \quad (3.33)$$

By Equations 3.30 and 3.33, we can obtain  $b'$  as follows:

$$b' = -\frac{\bar{U} - u_c}{\bar{V} - v_c}. \quad (3.34)$$

$$c' = \frac{\bar{U} - u_c}{\bar{V} - v_c} v_c - u_c \quad (3.35)$$



Therefore, we can represent each refitting line  $L$  by  $b'$  and  $c'$  described above. After such constrained line refitting is conducted, the precisions of the house corner edge lines are improved.

Finally, only two house corner edge lines are necessary in this study, and the unnecessary corner edge is the vertical line. Discarding of the vertical house corner edge can be accomplished by judging the slopes of the three refitted corner edge lines. An example is shown in Figure 3.14, and the color lines are non-vertical corner edge lines. By the two corner edge lines, the necessary coefficients are obtained. A detailed algorithm is described in the following



Figure 3.14 Refitting of two non-vertical corner edges.

**Algorithm 3.2.** *Constrained line refitting and determination of non-vertical lines.*

*Input:* A set of house corner edges described in terms of  $\alpha$  and  $\gamma$ .

*Output:* the coefficients of non-vertical house corner edges.

*Steps:*

- Step 1. Group together lines whose  $\alpha$  and  $\gamma$  are closer enough.
- Step 2. Transform lines described in terms of coordinates  $\alpha$  and  $\gamma$  into ones described in terms of image coordinates by Equation 3.26.
- Step 3. Compute the intersections of the lines in different groups by Cramer's rule.
- Step 4. Compute the mean of the intersections in terms of image coordinates and denote the result as  $P_{center}$ .
- Step 5. Collect edge line points in the same group into  $S$ .
- Step 6. Refit line  $L$  by Equations 3.33 and 3.34 using the data of  $S$  and  $P_{center}$ .
- Step 7. Determine the two non-vertical edges by the slopes of the three refitting edge line equations, with the slope of the vertical edge being taken to approach infinity.
- Step 8. Return the coefficients of two non-vertical edge line equations.



More experimental results of constrained line refitting are shown in Figure 3.15 and Figure 3.16.

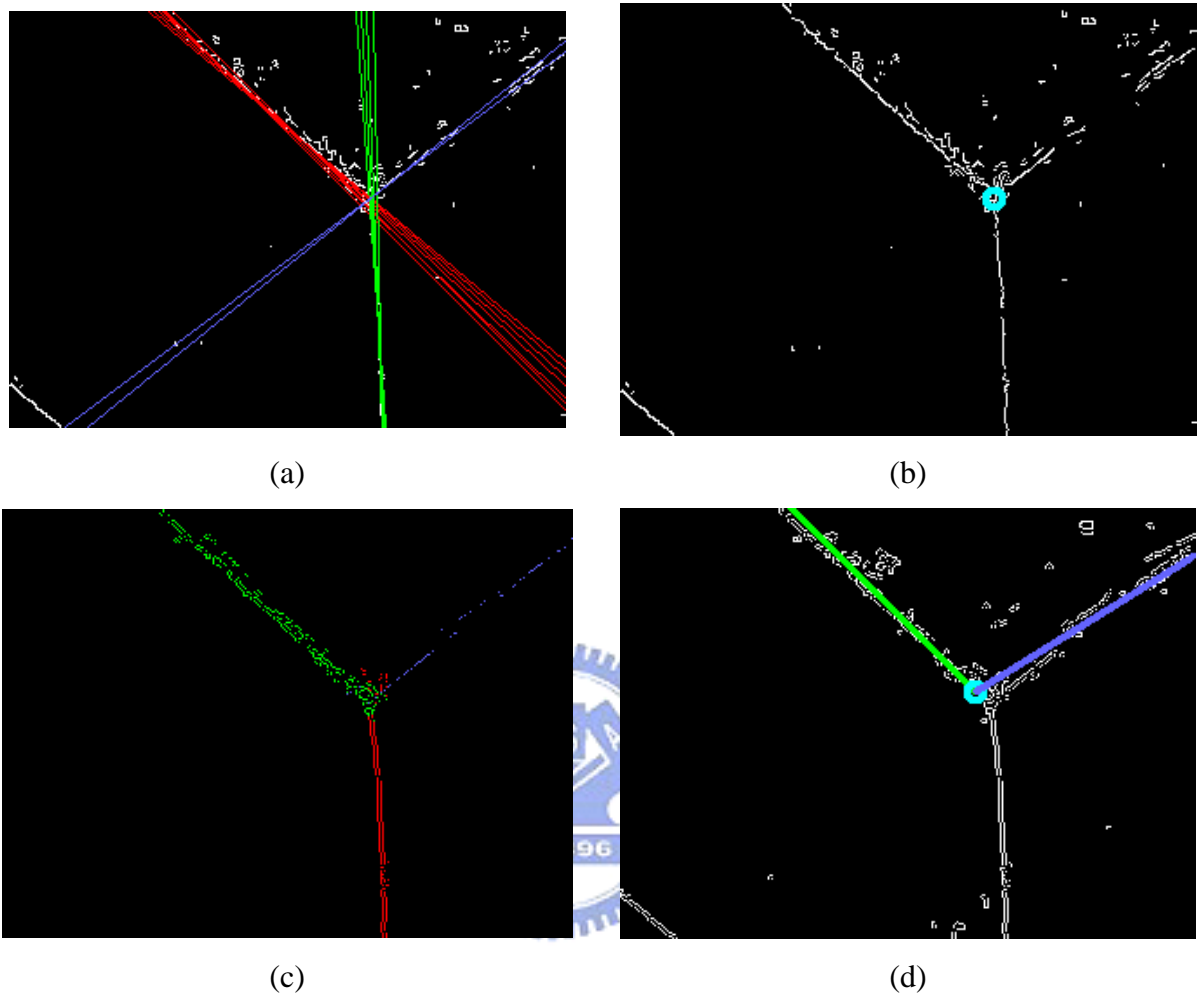


Figure 3.15 An experimental result of a different house corner. (a) The sets of lines represent an identical house corner edge. (b) The house corner center. (c) The edge points (d) Refitting of two non-vertical corner edges

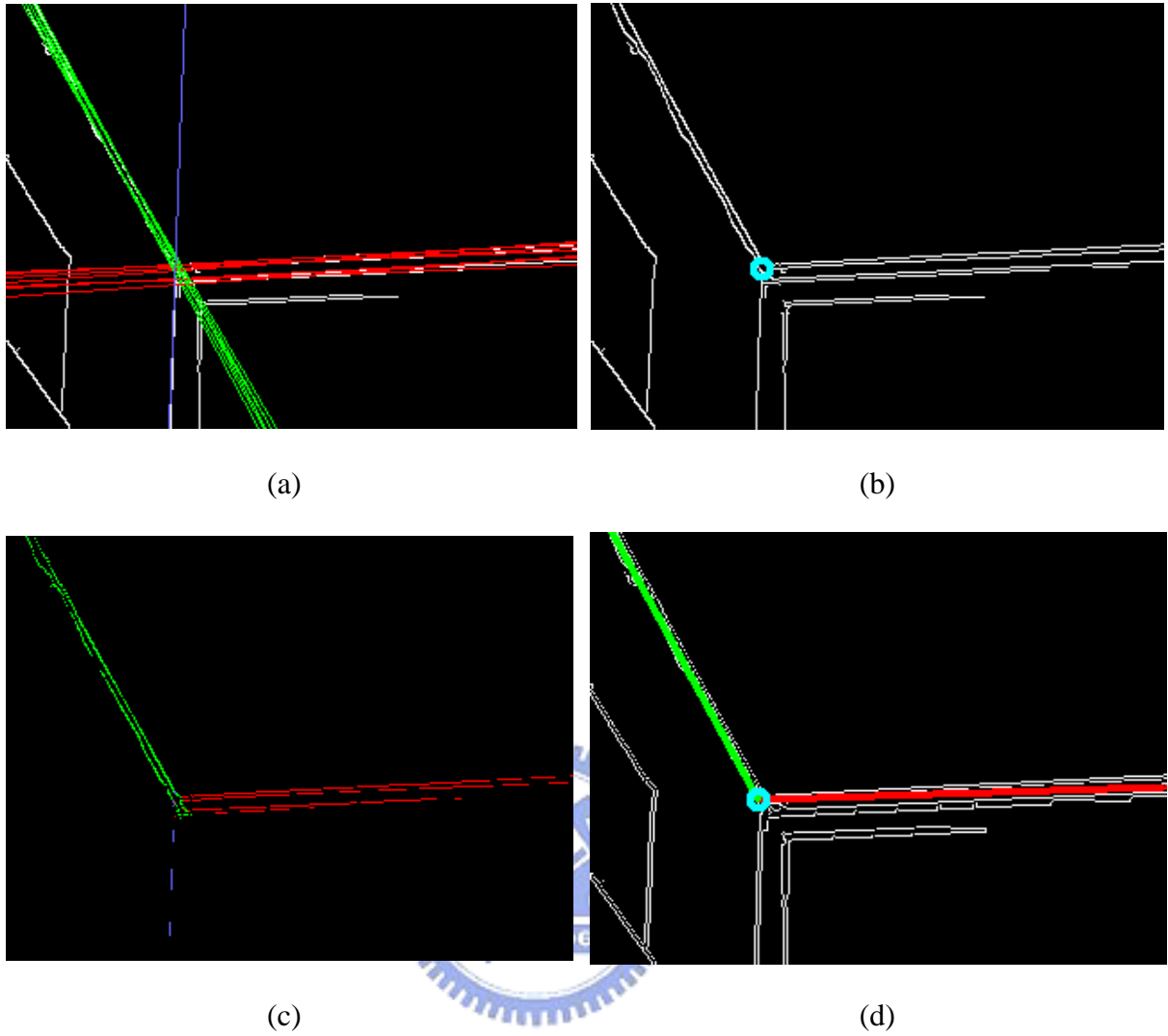


Figure 3.16 One more experimental result of another different house corner. (a) The sets of lines represent an identical house corner edge. (b) The house corner center. (c) The edge points (d) Refitting of two non-vertical corner edges

# Chapter 4

## Obstacle Avoidance by Goal-Directed Minimum-Path Following

### 4.1 Overview of Obstacle Avoidance

An obstacle appearing on navigation path suddenly is an ordinary situation in vehicle navigation. Many methods for obstacle avoidance have been proposed recently. Ku and Tsai [11] proposed a quadratic classifier for obstacle avoidance. Chen and Tsai [12] proposed a fuzzy guidance technique for the same purpose. However, most methods are specific in their applications. A general solution for obstacle avoidance is difficult. In order to avoid an obstacle in navigation, an obstacle avoidance method is necessary for our vehicle system. Because usually the navigation path is created in advance, the vehicle must avoid obstacles in real time. Therefore, detecting obstacles quickly is necessary to react to the situation.

The first step of obstacle avoidance is obstacle detection. The obstacle detection method used in this study is described in Section 4.2. If an obstacle is detected on the navigation path, a new path should be computed to avoid the obstacle. And the goal of the new path is toward the next node of the navigation path. The proposed process for this purpose is described in Section 4.3. The process of obstacle avoidance is illustrated in Figure 4.1.

In order to achieve the task, we need to know obstacle distributions in real space. Thus we define some coordinate systems and a direction angle of the vehicle in this

study at first. The coordinate systems are defined in Section 4.1.1 and the direction angle of the vehicle is defined in Section 4.1.2. By using the direction angle, transformations between these coordinate systems is possible. We utilize the transformation functions to obtain obstacle distributions in real space. The transformation function is described in Section 4.1.3.

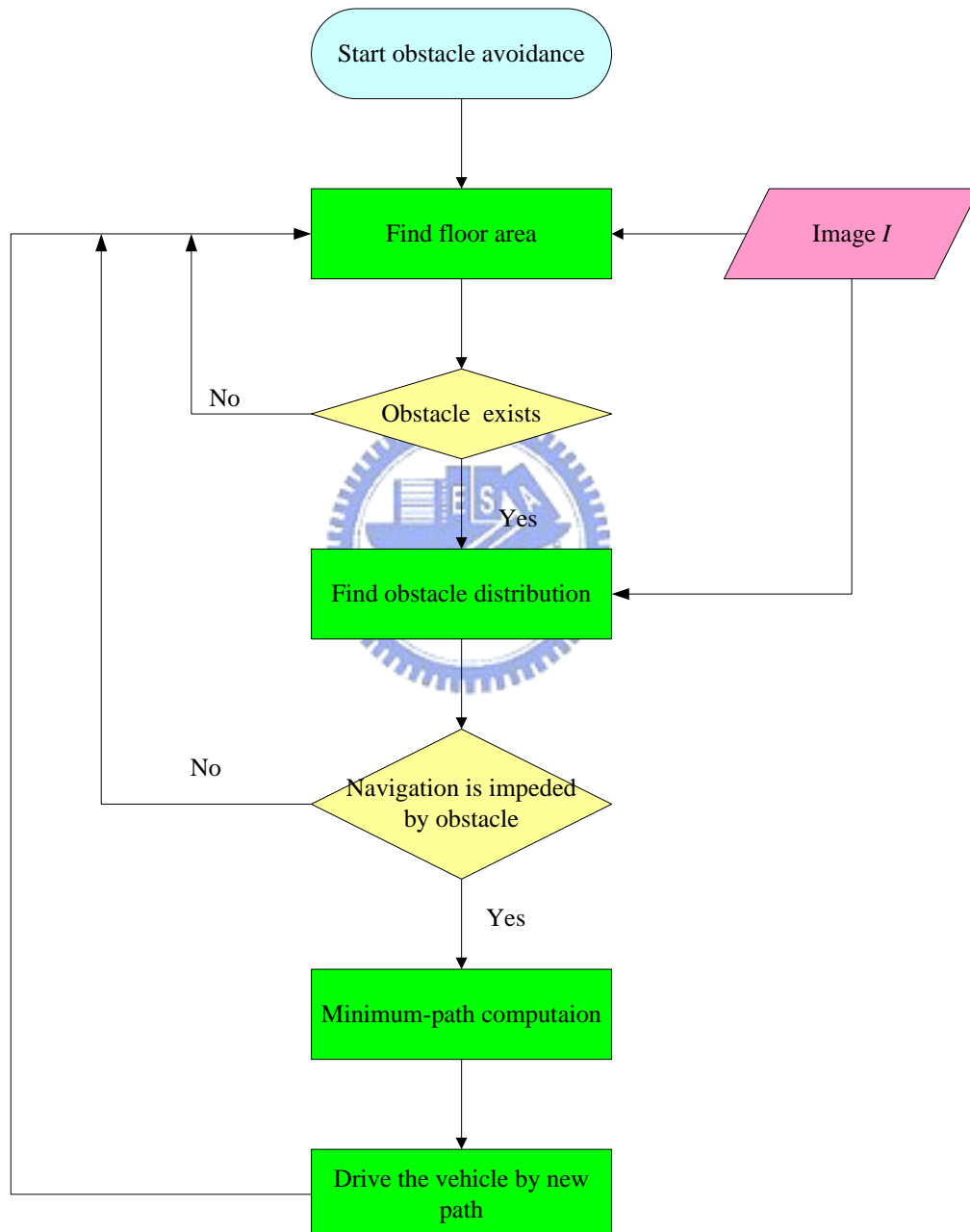


Figure 4.1 Flowchart of obstacle avoidance process.

## 4.1.1 Coordinate Systems

In this study, the following two coordinate systems are used to describe the vehicle location and the navigation environment. These coordinate systems are illustrated in Figure 4.2 and defined in the following.

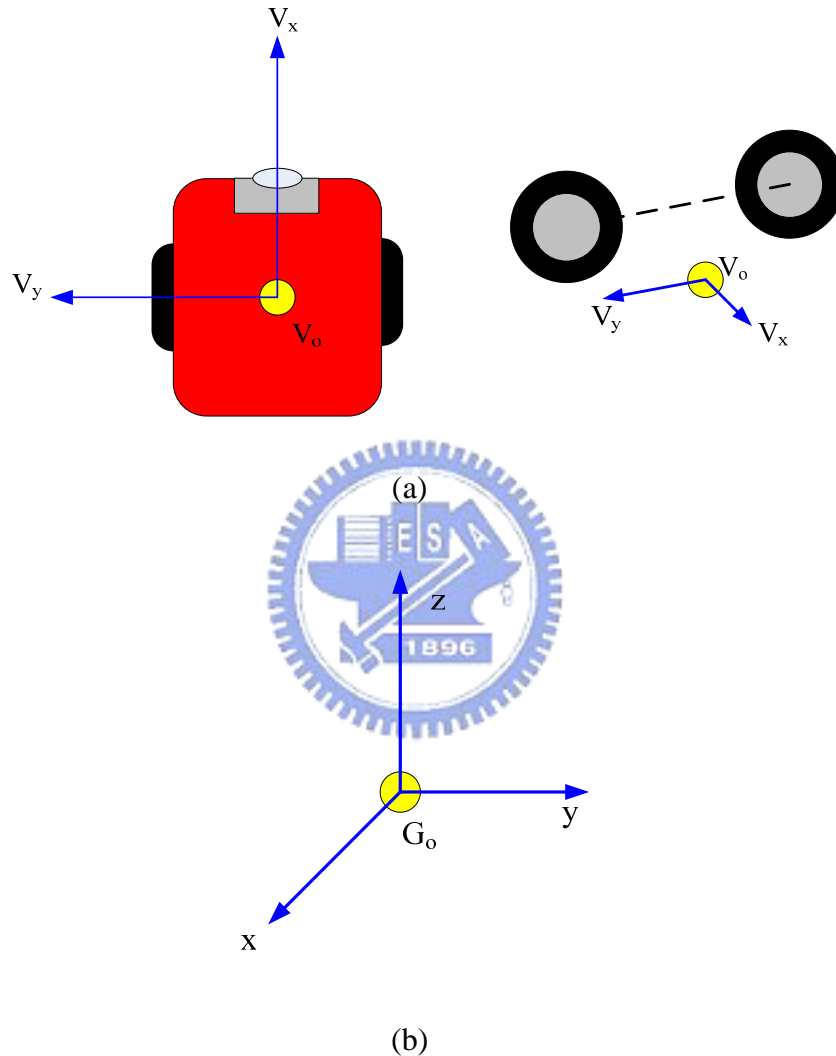


Figure 4.2 Two coordinate systems in this study. (a) The vehicle coordinate system.  
(b) The global coordinate system.

- (1) The vehicle coordinate system (VCS, denoted as  $V_x$ - $V_y$ ): The origin  $V_o$  of the vehicle coordinate system is placed at the middle point of the line segment which connects the two contact points of the two wheels with the ground. The  $V_x$  and  $V_y$  axes are on the ground. The  $V_x$ -axis is parallel to the body of the vehicle and the

$V_y$ -axis is perpendicular to the  $V_x$ -axis.

- (2) The world coordinate system (WCS, denoted as  $x$ - $y$ ): The origin  $G_o$  of the world coordinate system is located at a certain fixed position and it is the starting position of each navigation session in this study. The  $x$ -axis and the  $y$ -axis are defined to lie to on the ground and perpendicular to each other.

## 4.1.2 Direction Angle of Vehicle

We need to know the state of the vehicle heading, so we define the direction angle of the vehicle in the world coordinate system. The angle, denoted as  $\omega$ , represents the rotation degree of the vehicle in the world coordinate system and plays an important role in coordinate transformation.

$\omega$  is the angle between the positive direction of the  $x$ -axis and the front of the vehicle. The direction angle  $\omega$  is set to be zero at the beginning of navigation. The range of  $\omega$  is between 0 and  $\pi$  if  $\omega$  is in the first and second quadrants, as illustrated in Figure 4.3(a). It is between 0 and  $-\pi$  if  $\omega$  is in the third and fourth quadrants, as illustrated in Figure 4.3(b).

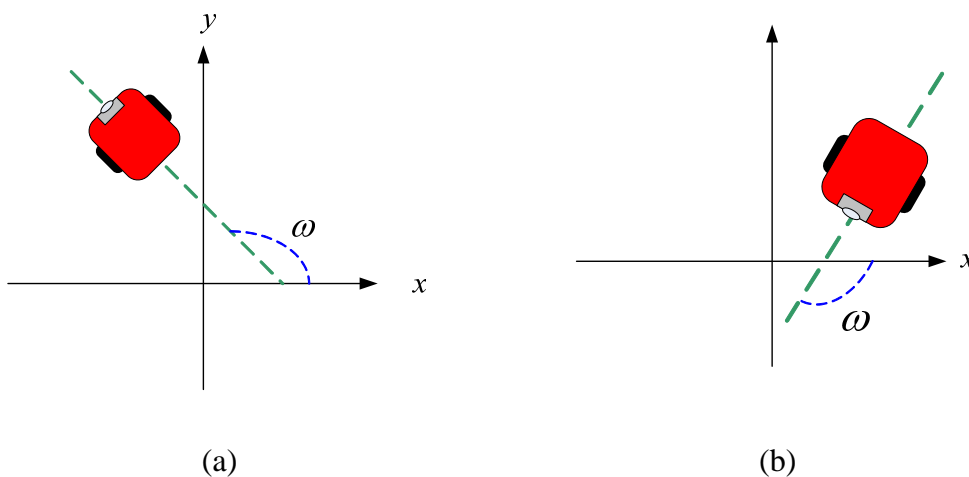


Figure 4.3 Definition of the direction angle (a)  $\omega$  is positive. (b)  $\omega$  is negative.

### 4.1.3 Coordinate Transformation

The transformation functions between the vehicle coordinate system and the world coordinate system using the direction angle is shown in Equation (4.1) and Equation (4.2) below, where  $x_p$  and  $y_p$  are the coordinates of the vehicle in the world coordinate system.

$$x = V_x \sin \omega - V_y \cos \omega + x_p \quad (4.1)$$

$$y = V_x \cos \omega + V_y \sin \omega + y_p \quad (4.2)$$

The coordinate transformation between the vehicle coordinate system and the world coordinate system is illustrated in Figure 4.4.

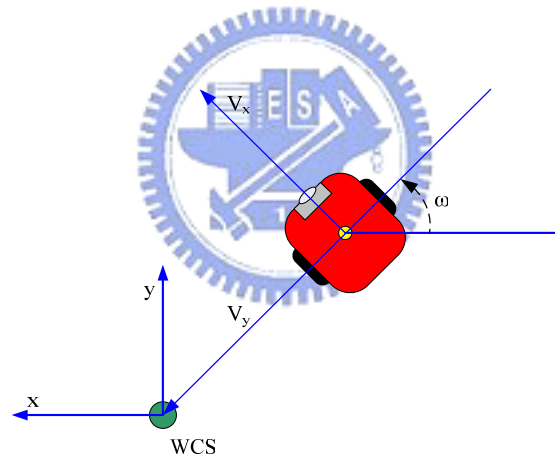


Figure 4.4 The coordinate transformation between the vehicle coordinate system and vehicle coordinate system.

## 4.2 Obstacle Detection Process

Overall, before the obstacle avoidance process is started, a process to detect obstacles is applied first. In this study, we observe the floor continuously to obtain the floor area all the time. Finding the floor area is important and convenient for detection

of obstacles on the navigation path. The process to find an obstacle is described in Section 4.2.1. As long as we obtain the floor area, the floor in the image can be filtered out. Therefore, we can deal with a new image without the floor area. We define a *warning area* for obstacle alert at first. Once we detect anything in the area, it means that something extraordinary is on the floor. The process for determination of obstacle existence is described in Section 4.2.2. A flowchart of the obstacle detection process is illustrated in Figure 4.5.

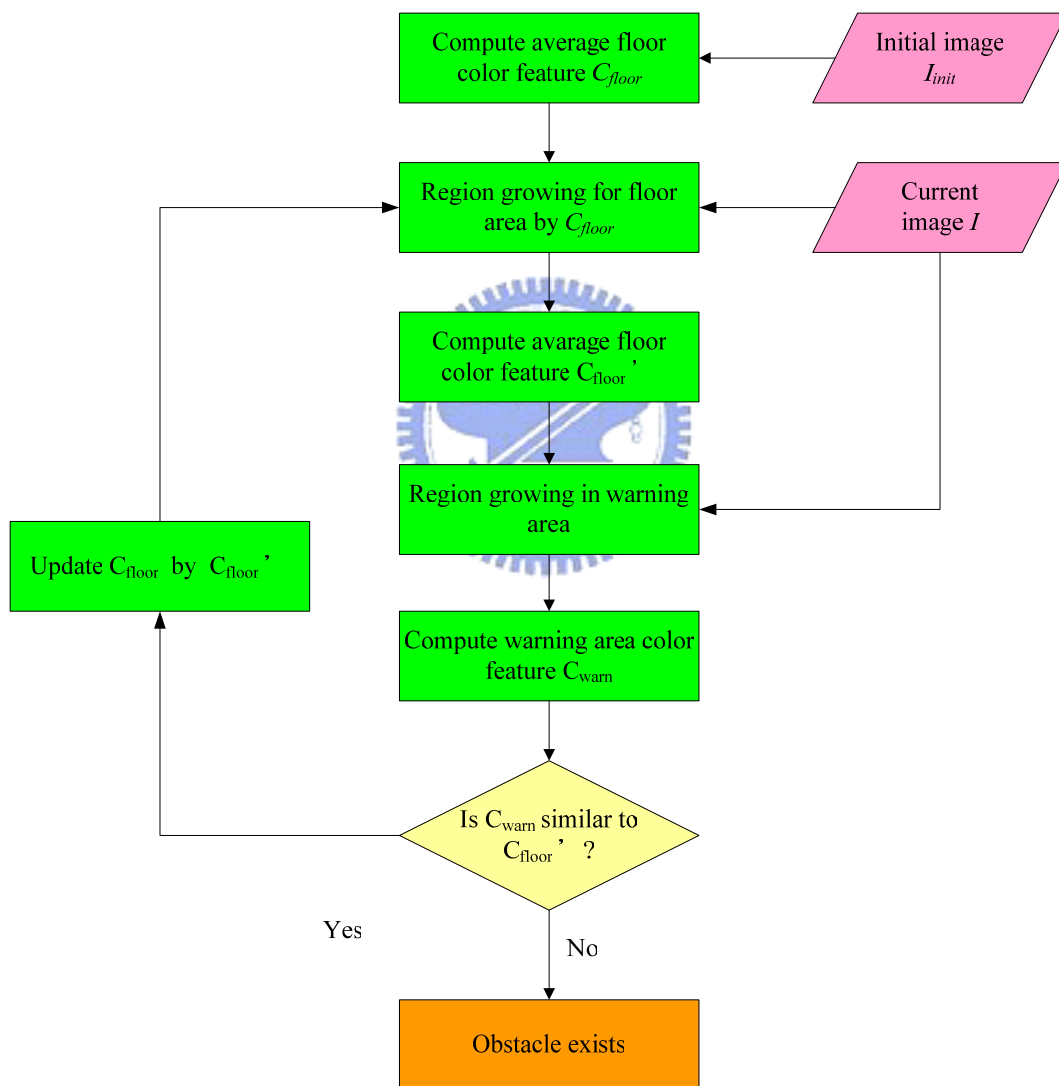


Figure 4.5 Flowchart of obstacle detection process.



## 4.2.1 Finding Floor Area

In order to find an obstacle on the floor, the first thing is to find the floor area in that image. One reasonable assumption is that there is no obstacle on the uniform floor in the initial image captured by the camera. Under this assumption, we can apply a region growing algorithm by an appropriate seed to find the floor area in the initial state. After finding the floor, we can retrieve the color feature of the floor and denote it as  $C_{floor}$ . The color feature here is computed in terms of the  $R$ ,  $G$ , and  $B$  color values. The definition is

$$C_{floor} = \left( \frac{\sum_{i=1}^n R_i}{n}, \frac{\sum_{i=1}^n G_i}{n}, \frac{\sum_{i=1}^n B_i}{n} \right)$$

where  $R_i$ ,  $G_i$ ,  $B_i$  are the R, G, and B values of a pixel  $P_i$  in the image, and  $n$  is the number of the floor pixels.

The color feature of floor is useful in determination of obstacle existence and finding a collision-free path later. In the first cycle, we perform a region growing algorithm with the initial color feature  $C_{floor}$  as a seed to find the floor area and record the new color feature of the floor area as  $C_{floor}'$ . At the end of the process, we update  $C_{floor}$  to be  $C_{floor}'$  for the next cycle. So in each cycle, we use the color feature of the floor of the last navigation cycle to find the floor area of the current navigation cycle. A floor area found by this method is shown in Figure 4.6. The algorithm is described in detail as follows.

**Algorithm 4.1** *Finding floor area.*

*Input:* A color image  $I$ .

*Output:* The floor area  $AREA_{floor}$  and the color feature  $C_{floor}$  of the floor area.

*Steps:*

- Step 5. Apply region growing with an appropriate seed to find the floor area  $I_{init}$  in the initial image  $I$ .
- Step 6. Compute the color feature of the floor area in  $I_{init}$ . Let the color feature be denoted as  $C_{floor}$ .
- Step 7. Capture a new Image of the latest situation of the environment and denote it as  $I_{new}$ .
- Step 8. Apply region growing with color feature  $C_{floor}$  in the image  $I_{new}$  to find the latest floor area  $AREA_{floor}$ .
- Step 9. Compute the color feature  $C_{floor}'$  of the floor area in  $I_{now}$ . Update the color feature  $C_{floor}$  to be the new one  $C_{floor}'$  for the next cycle of finding the floor area.

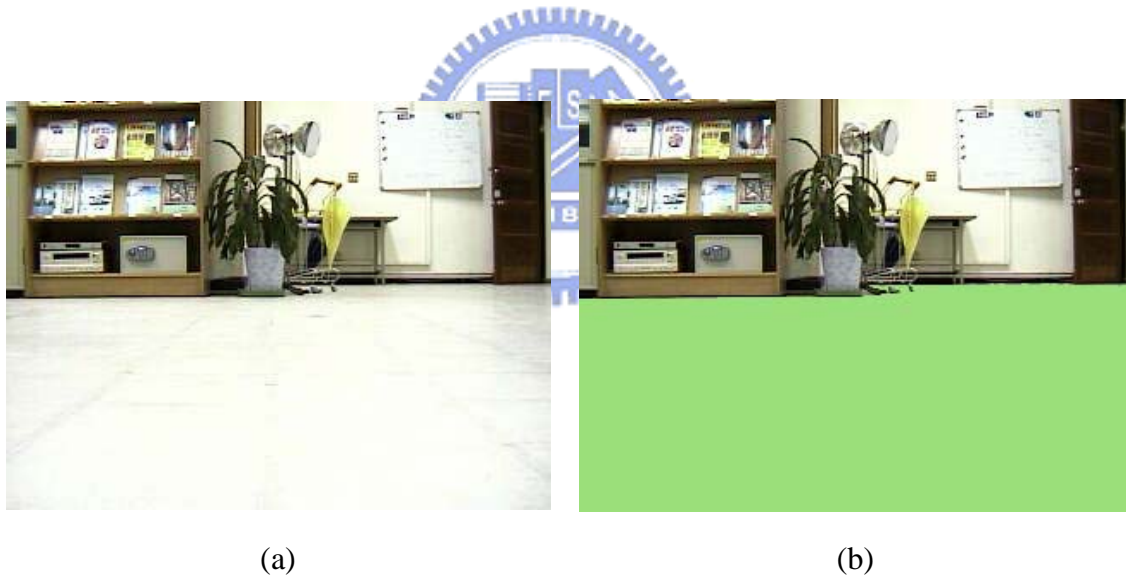


Figure 4.6 Region growing to find the floor area in an image. (a) The floor image. (b) Detection of the floor area.

## 4.2.2 Determination of Obstacle Existence

Once we own the color feature of the floor, the detection of an obstacle becomes easier. We define a *warning area* in an image in advance to detect obstacles. This area is established by two parallel horizontal lines in two different fixed lower parts of the

image. If an obstacle is detected in the area, it means that the vehicle should change its navigation path to avoid the obstacle.

We scan the image from bottom to top and left to right to find the region which is not part of the floor. Then, we compute the color feature of the region in the same form of  $C_{floor}$ . If the  $C_{warn}$  is similar to  $C_{floor}$ , it means that the region is also the floor area. Otherwise, the region represents an obstacle. Figure 4.7 shows an obstacle on the floor and the result after obstacle detection. An algorithm for the detection of obstacles is described in detail as follows.

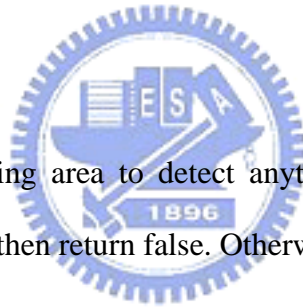
**Algorithm 4.2** *Determination of an obstacle existence.*

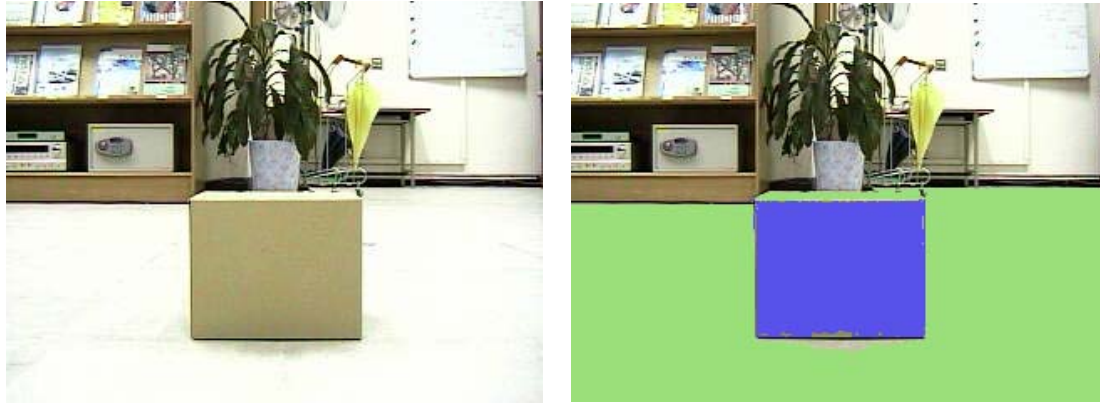
*Input:* An image with an obstacle and the floor area.

*Output:* A Boolean value, true or false, meaning the existence and the reverse of an object in the campus.

*Steps:*

- Step 1. Search in the warning area to detect anything existing on the floor. If nothing is detected, then return false. Otherwise, compute the color feature  $C_{warn}$  of the region.
- Step 2. Compare the difference between  $C_{warn}$  and  $C_{floor}'$ . Denote the difference as  $Diff$ .
- Step 3. If  $Diff$  is greater than a threshold and the region is larger enough, then an obstacle exists. We return true. Otherwise, we return false.





(a)

(b)

Figure 4.7 Detection of an obstacle within the warning area. (a) An image including an obstacle. (b) The detection of the obstacle.

## 4.3 Minimum-Path Following Process

Once an obstacle is detected, a new path for achieving obstacle avoidance is necessary. We accomplish such a task by inserting a new node into the navigation map to modify the navigation path. The method to compute such a new node we propose in this study is described in Section 4.3.2. Before we compute the new node, we have to detect the *obstacle distribution* in the world coordinate system. The process is described in Section 4.3.1.

### 4.3.1 Detection of Obstacle Distribution

After we detect an obstacle on a patrolling path, we judge first the size of the obstacle. If the size of the obstacle is not large enough, it may be a mistake or it does not affect the navigation. If the result indicates that the obstacle is large enough to affect navigation, we have to compute the obstacle distribution in the world coordinate system.

Because the view of the vehicle is monocular, we cannot obtain the depth of the obstacle. However, we can still find the projection of the obstacle on the floor from an image. Thus, we define *the projection of the obstacle distribution* as the projection of the front of the obstacle on the floor in the image coordinate system. We can find the projection of the obstacle distribution in the image coordinate system by scanning the vertical lines from the bottom of the image to the top of the image.

Once we know the projection of the obstacle distribution, we can transform the obstacle in the image coordinate system into that in the vehicle coordinate system according to an interpolation method [5]. We map the tessellated points in the image coordinate system to the ones in the world coordinate system in advanced. The tessellated point illustration is shown in Figure 4.8. The interpolation method is using this pre-defined 2D mapping from image coordinate system to the vehicle coordinate system and any point that is not at the tessellated point exactly is computed by interpolation method. After we obtain the obstacle distribution in the vehicle coordinate system, we use Equations 4.1 and 4.2 to transform the obstacle in the vehicle coordinate system into that in the world coordinate system. In this way, we obtain the front of the obstacle in the world coordinate system. An example is illustrated in Figure 4.9. A detailed algorithm is described in the following.

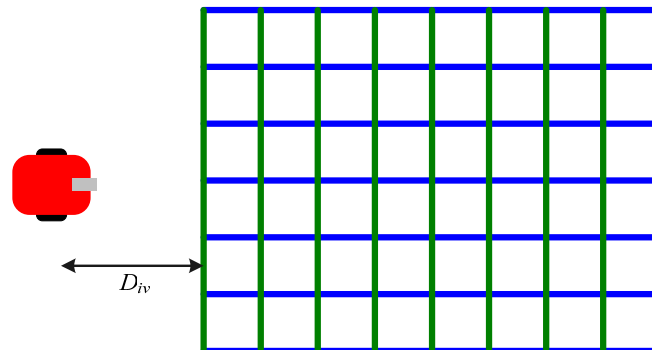


Figure 4.8 An illustrated of attaching the lines on the floor.

**Algorithm 4.3** Finding an obstacle distribution.

*Input:* An image with an obstacle.

*Output:* The front of the obstacle in the WCS.

*Steps:*

- Step 1. Find the projection of the obstacle distribution in the ICS.
- Step 2. Transform the projection of the obstacle distribution from the ICS to the VCS by an interpolation method to obtain the obstacle distribution in VCS.
- Step 3. Transform the obstacle distribution from the VCS to the GCS by Equations 4.1 and 4.2.
- Step 4. After Step 3, the front of the obstacle in the WCS is obtained.

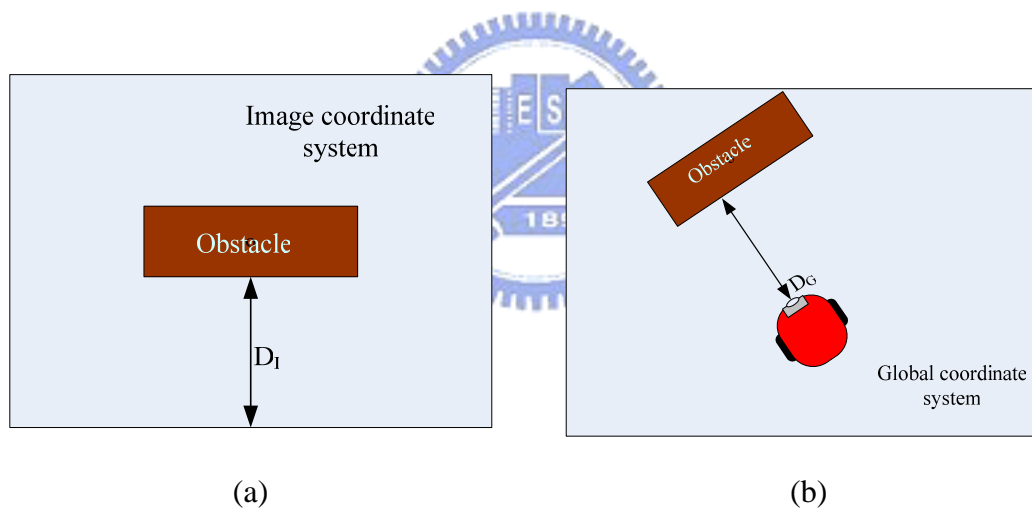


Figure 4.9 Using an obstacle image to find the front of the obstacle in the WCS. (a) An obstacle in image coordinate system. (b) The front of the obstacle in world coordinate system.

### 4.3.2 Computation of Goal-Direction Minimum Path

After an obstacle is confirmed to affect navigation, a new path is planned to guide the vehicle to avoid the obstacle. In this study, because our guidance principle is moving sequentially from one node to another according to the navigation map, we can change the navigation path by modify the node sequence. In other words, if we

insert a new node before the vehicle reaches the next node, the vehicle will change its navigation path toward the new node. The result is that the vehicle will move to the new node first and go to the original node in the next step.

Under this guidance principle, we can insert a node into the navigation map to guide the vehicle to avoid an obstacle. The key point is how to decide such a node that achieves the purpose. And this node will not guide the vehicle move too far from the next node. Thus, we propose a method to compute a node for goal-direction minimum-path guidance in Algorithm 4.4.

The idea is that we deal with an obstacle once. At first, we transform the obstacle in the image coordinate system to the vehicle coordinate system, so we know the relationship of the obstacle and the vehicle. The computation of a goal-direction minimum path is illustrated in Figure 4.10, where  $W_V$  is the appropriate width for the vehicle to pass and  $D_G$  is the shortest distance between the obstacle and the vehicle.

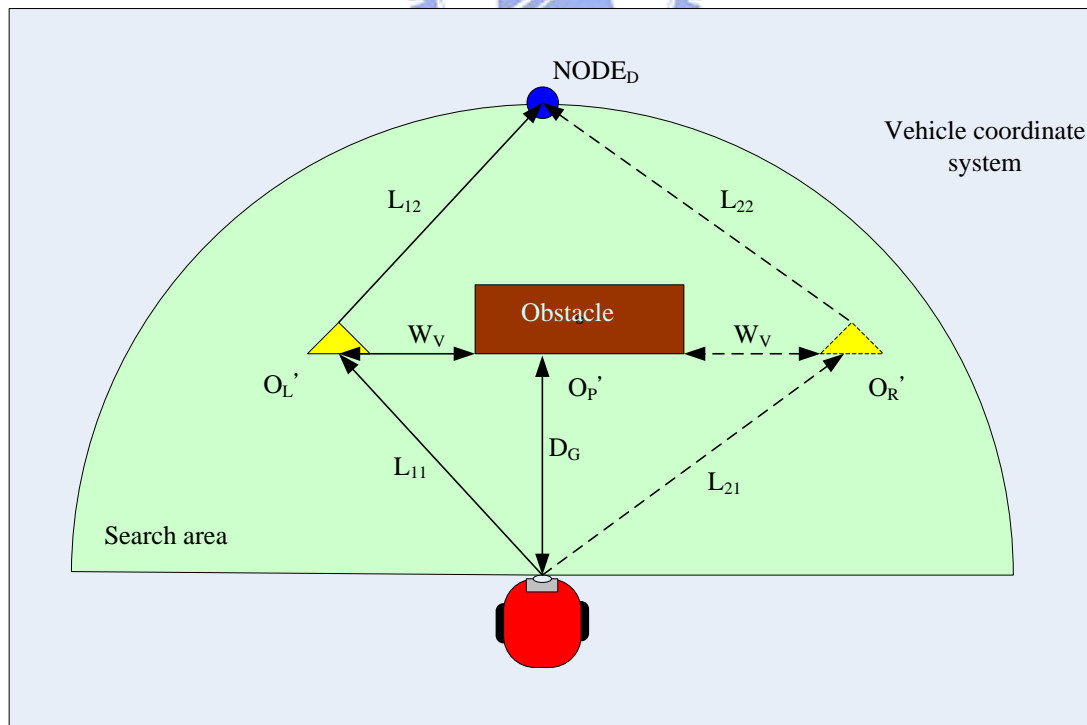


Figure 4.10 An illustration of computation of goal-direction minimum path.

We know the front of the obstacle in the vehicle coordinate system, so we can find a point  $O_P$  of the obstacle which is closest to the vehicle. We also know the left and the right end sides of the obstacle in the vehicle coordinate system and denote them as  $O_L$  and  $O_R$ . Thus we can obtain two obstacle avoidance nodes in the vehicle coordinate system in the following:

$$O_L' = \vec{u} \cdot \vec{W}_v + \overrightarrow{O_L O_P} + \overrightarrow{O_P V_P}; \quad (4.3)$$

$$O_R' = \vec{u} \cdot \vec{W}_v + \overrightarrow{O_R O_P} + \overrightarrow{O_P V_P}, \quad (4.4)$$

where  $O_L'$  and  $O_R'$  are the new obstacle avoidance nodes,  $\vec{u}$  is the unit vector in the vehicle coordinate system, and  $V_P$  is the vehicle position.

Now, we transform  $O_L'$  and  $O_R'$  from the vehicle coordinate system to the world coordinate system by Equations 4.1 and 4.2. Then we can obtain the left and the right end points of the obstacle in the world coordinates. We denote the left point as  $O_{WL}$  and the right one as  $O_{WR}$ . Thus, the length of each path can be computed in the following way:

$$Length_L = \left| \overrightarrow{V_W O_{WL}} \right| + \left| \overrightarrow{O_{WL} N_{next}} \right|; \quad (4.5)$$

$$Length_R = \left| \overrightarrow{V_W O_{WR}} \right| + \left| \overrightarrow{O_{WR} N_{next}} \right|, \quad (4.6)$$

where  $Length_L$  is the length of the path after inserting the node  $O_{WL}$ ,  $Length_R$  is the length of the path after inserting the node  $O_{WR}$ ,  $V_W$  is the vehicle location in the GCS, and  $N_{next}$  is the position of the next node.

We compare  $Length_L$  and  $Length_R$  to decide which node should be used for the goal-direction minimum path. We also set a search area which is a semicircle with the radius being the length between the vehicle location and the next node. This search area can guarantee that the new node will not guide the vehicle too far from the next



node by limiting the node in the search area. We use the following formula:

$$\text{If } |\overrightarrow{V_w N}| > r \text{ then } N = u_N \cdot r, \quad (4.7)$$

where  $N$  is the node for obstacle avoidance,  $r$  is the distance between the vehicle and the next node, and  $u_N$  is the unit vector of  $\overrightarrow{V_w N}$ .

In the next cycle, we perform the whole process again to detect and avoid an obstacle. Thus the vehicle can avoid multi-obstacles continuously. Figure 4.11 shows the top view of the obstacle avoidance process.



Figure 4.11 Top view of an obstacle avoidance experiment.

**Algorithm 4.4** *Computation of goal-direction minimum path.*

*Input:* The obstacle distribution of the front of the obstacle in the VCS.

*Output:* A node of goal-direction minimum path.

*Steps:*

- Step 1. Compute the two candidate nodes  $O_L'$  and  $O_R'$  for obstacle avoidance in the VCS by Equations 4.3 and 4.4.
- Step 2. Compute the two lengths  $Length_L$  and  $Length_R$  for obstacle avoidance with  $O_L'$  and  $O_R'$  by Equations 4.5 and 4.6
- Step 3. If  $Length_L$  is smaller than  $Length_R$ , we choose  $O_L'$  as the obstacle

avoidance node; else, we choose  $O_R'$ .

Step 4. Using Equation 4.7 to check whether the node is in the search area or not.

If it is not, compute the node position by Equation 4.7.

Step 5. Return the node position.



# Chapter 5

## Vision-Based Indoor Danger Condition Monitoring

### 5.1 Principle of Danger Condition Monitoring

One important issue in a vehicle security surveillance system is danger condition monitoring. The earlier danger condition is detected, the less damage is made. It also prevents the danger condition from getting worse. Therefore, to detect danger conditions quickly and correctly is a main issue in this study. However, to detect quickly and correctly is not always a good policy. If the detection is more precise, the computation complication will increase. Once the computation complication increases, the spent time will increase, too. In real time applications, it may lose the best chance easily in time to take some remediable actions.

In Section 5.2, we will discuss the difference between the RGB color model and the HSI color model. The reason for using the HSI color model instead of the RGB color model is also described. In Section 5.3, we discuss the dangerous fire condition and the method we propose to detect a fire. In Section 5.4, we discuss the lighting failure condition, and the method we propose to detect lighting failure.

## 5.2 Advantage of Using HSI Color Model

The RGB color model presents each color in its primary spectral components of red, green, and blue. A 24-bit RGB color cube is shown in Figure 5.1. A particular color is represented by the amount of each of the primary components. The RGB model simplifies the design of computer graphics systems and is used for color monitors and most video cameras. But it is not ideal for all applications, because the red, green, and blue color components are highly correlated. This makes it difficult to execute some image processing algorithms. Many processing techniques work better on the intensity component of an image. These processes are used more easily and work more efficiently in the HSI color model.

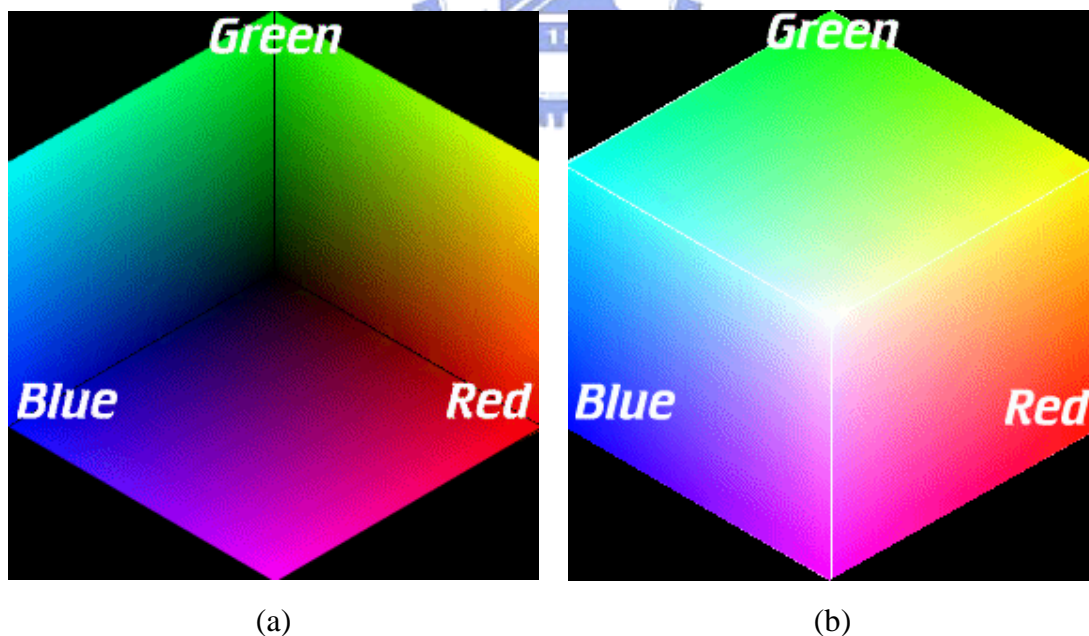


Figure 5.1 RGB 24-bit color cube. (a) Cube with black corner. (c) Cube with white corner [13].

The HSI color model is one kind of color models describing a color object by its hue, saturation, and intensity. This color model is important and attractive because it represents colors similarly to the way the human eye senses colors. Hue is a color attribute that describes a pure color (pure yellow, orange, or red), whereas saturation gives a measure of a degree to which a pure color is diluted by white light. Intensity is a subjective descriptor [13]. Figure 5.2 shows the HSI color model based on color circles. The hue component contains the color information of the R, G, and B components. The HSI color model decouples the intensity component from the color-carrying information (hue and saturation) in a color image. So the HSI color model allows independent control over hue, saturation, and intensity.

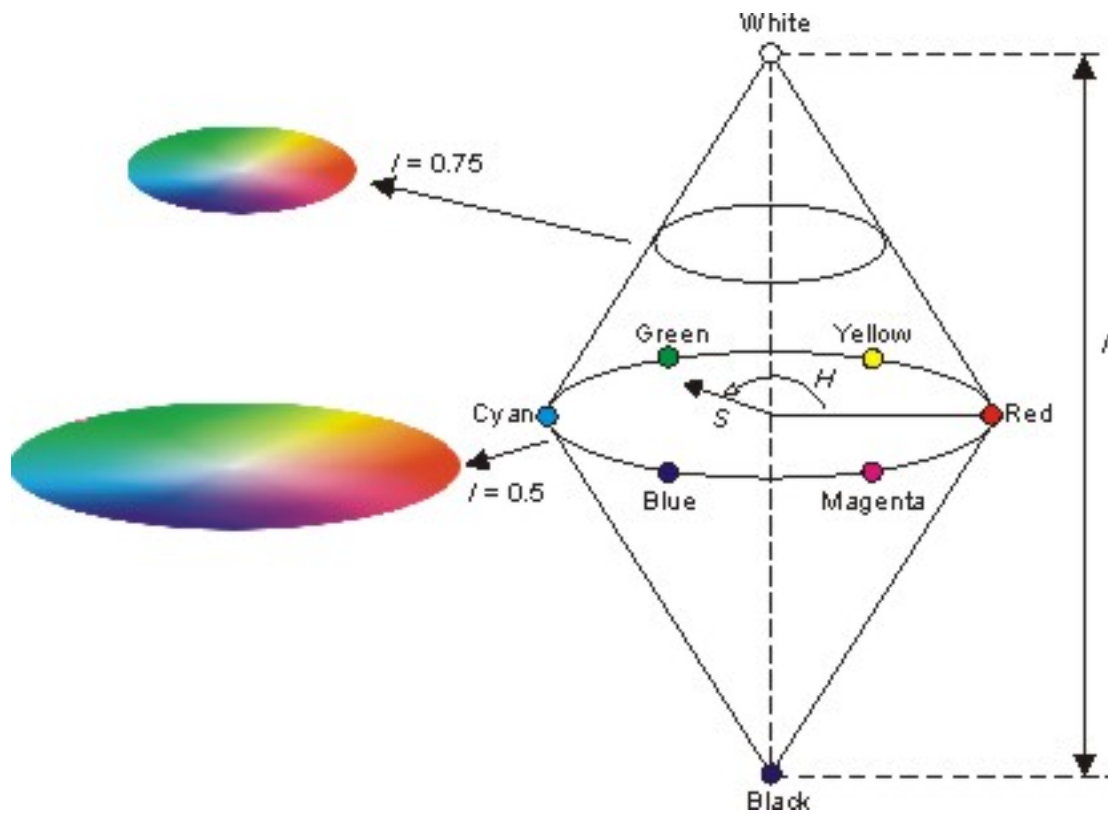


Figure 5.2 The HSI color model based on circular color planes. The circles are perpendicular to the vertical intensity axis [13].

As mentioned before, the RGB color model is ideal for a color camera. The camera used in this study also acquires images in the RGB color model. So we need

to transform the RGB color model into the HSI color model by the following formulas:

$$\begin{cases} H = \begin{cases} \theta & \text{if } B \leq G; \\ 360 - \theta & \text{if } B > G; \end{cases} \\ \theta = \cos^{-1} \left\{ \frac{\frac{1}{2}[(R-G) + (R-B)]}{\left[ \frac{1}{4}[(R-G)^2 + (R-B)(G-B)] \right]^{\frac{1}{2}}} \right\}; \\ S = \left\{ 1 - \frac{3}{(R+G+B)} [\min(R, G, B)] \right\} \times 180; \\ I = \frac{1}{3}(R+G+B). \end{cases} \quad (5.1)$$

## 5.3 Dangerous Fire Condition

Fire is one of the most threatening danger conditions in the world. Detecting fire earlier reduces the damage. Most existing methods for fire detection are based on uses of infrared and ultraviolet sensors. A drawback is that sensors must be installed near the place where fire might be caused, or it will not be effective to detect the fire. Another drawback is that it can monitor only the place where sensors are installed. In this study, we use a movable vehicle equipped with a camera to perform vision-based fire detection. The vehicle can patrol everywhere in indoor environments to prevent harm caused by fire. The characteristics of fire images are described in Section 5.3.1, and the method proposed for fire detection is described in Section 5.3.2.

### 5.3.1 Fire Image Features

One difficulty in detecting fire is that fire does not possess a specific shape. It leads to the fact that detecting fire by pattern is hard to carry out. Fortunately, fire

images usually present yellow and red colors. High brightness is also an obvious and important feature of a fire image. We may analyze a lot of images with fire, and check the distribution and histogram of the hue, saturation, and intensity features. The red and yellow colors of fire in images are usually found within some ranges of saturation. We denote the ranges as  $H_{f1}$  and  $H_{f2}$  which are from 0 degree to 45 degrees and from 246 degrees to 338 degrees, respectively. The value of normalized saturation values of fire is larger than 0.1, which we denote as  $S_f$ . The normalized value of the intensity of fire in images is almost 1 and we denote it as  $I_f$ .

### 5.3.2 Proposed Fire Detection Method

While the vehicle patrols in indoor environments, the camera captures images continuously. Each image provides the situation at the time image taking. We analyze the image to judge whether fire exists or not.

In order to detect fire, we need the hue, saturation, and intensity information in the image. We transform the image from the RGB color model to the HSI color model. The HSI color model is convenient for us to retrieve the information about hue, saturation, and intensity. Then, we filter the images by certain thresholds of hue, saturation, and intensity. On the other hand, we use  $H_{f1}$ ,  $H_{f2}$ ,  $S_f$ , and  $I_f$  to keep necessary components in the image. After the process, we can get three filtered images which fit in with fire. We overlap the three images and keep the intersection to guarantee that the remainders represent fire. Finally, we count the non-black pixels in the result. If the number of such pixels is great enough, it means that a fire is detected. The complete fire detection process is described in following algorithm and a detailed flowchart is illustrated in Figure 5.3. The result of an image without fire is shown in Figure 5.4. On the contrary, the result of an image with fire is shown in Figure 5.5.

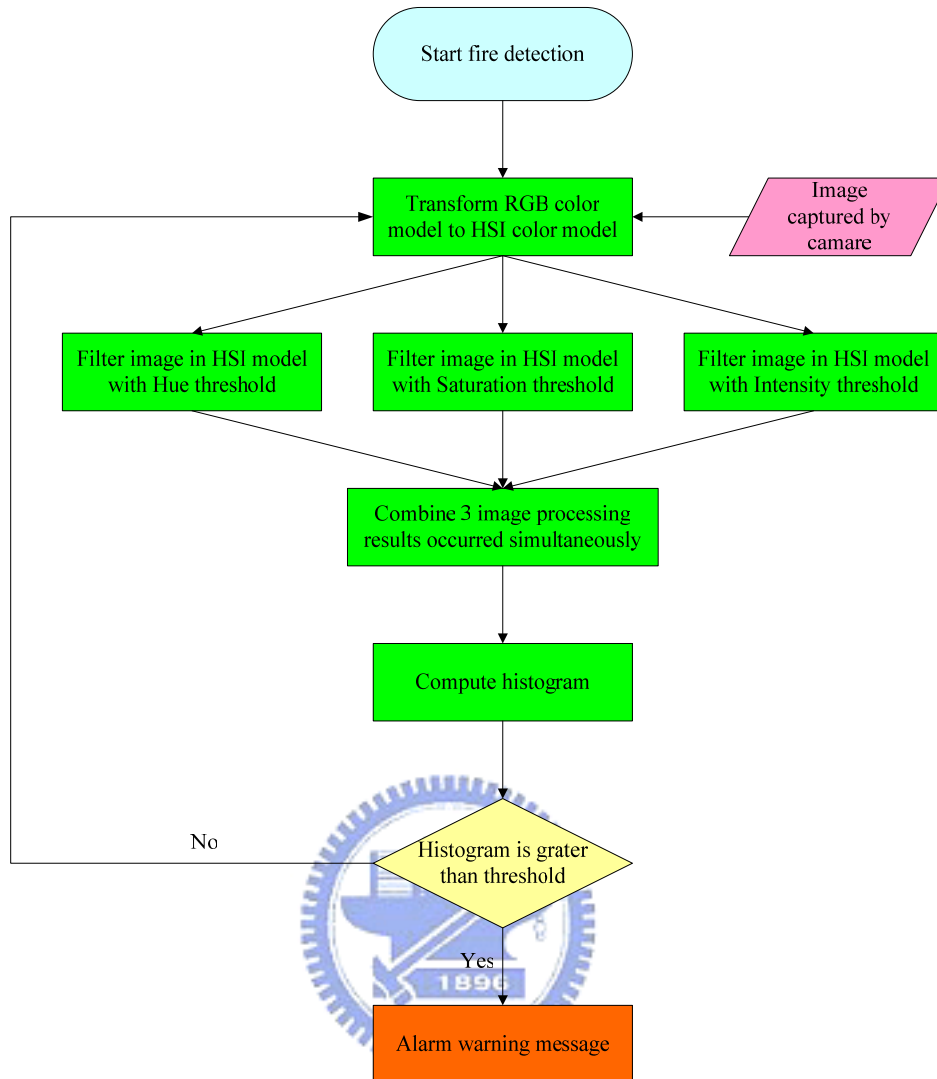


Figure 5.3 Flowchart of fire detection.

**Algorithm 5.1** Fire detection.

*Input:* A color image  $I$ .

*Output:* A fire warning message or nothing

*Steps:*

Step 1. Transform the image  $I$  in the RGB color model to the HSI color model by Equation 5.1. The hue is represented by  $I_H$ , the saturation by  $I_S$ , and the intensity by  $I_I$ .

Step 2. Keep the pixels in image  $I_H$ ,  $I_S$ , and  $I_I$  according to the thresholds  $H_{f1}$ ,  $H_{f2}$ ,  $S_f$ , and  $I_f$  in the following way.



- 2.1 Keep the pixels in the image  $I_H$  which are among  $H_{f1}$  and  $H_{f2}$ . Denote the reserved ones as  $I_H'$ .
  - 2.2 Keep the pixels in the image  $I_S$  which are greater than  $S_f$ . Denote the reserved ones as  $I_S'$ .
  - 2.3 Keep the pixels in the image  $I_I$  which are greater than  $Th_I$ . Denote the reserved ones as  $I_I'$ .
- Step 3. Overlap the images  $I_H'$ ,  $I_S'$ , and  $I_I'$  to obtain the intersection components, and denote the result as  $I_{inter}$ .
- Step 4. Compute the non-black pixels in  $I_{inter}$ . Denote the result as  $P_{num}$ .
- Step 5. If  $P_{num}$  is greater enough, then show a warning message.



(a)

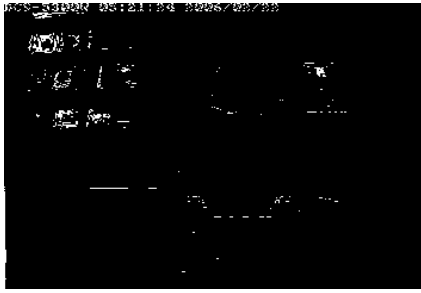


(b)



(c)

Figure 5.4 (a) An image without fire. (b) The hue of (a) within the ranges  $H_{f1}$  or  $H_{f2}$ .  
(c) The saturation of (a) over than  $S_f$ . (d) The intensity of (a) equals  $I_f$ . (e)  
The intersection of (b), (c), (d), and (e)



(d)

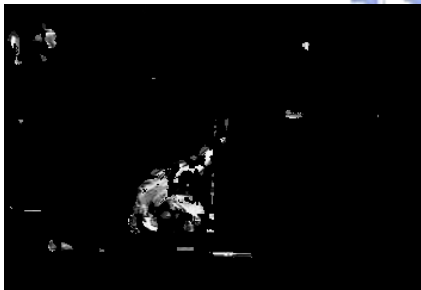


(e)

Figure 5.4 (a) An image without fire. (b) The hue of (a) within the ranges  $H_{f1}$  or  $H_{f2}$ . (c) The saturation of (a) over than  $S_f$ . (d) The intensity of (a) equals  $I_f$ . (e) The intersection of (b), (c), (d), and (e).



(a)



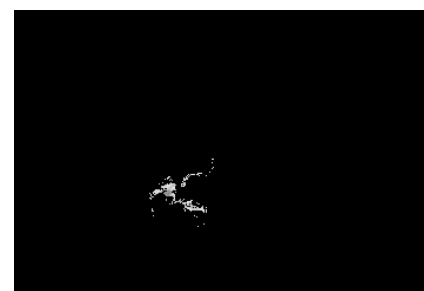
(b)



(c)



(d)



(e)

Figure 5.5 (a) An image with fire. (b) The hue of (a) within the ranges  $H_{f1}$  or  $H_{f2}$ . (c) The saturation of (a) over than  $S_f$ . (d) The intensity of (a) equals  $I_f$ . (e) The intersection of (b), (c), (d), and (e).

## 5.4 Lighting Failure Condition

The lighting failure is also a common danger condition in daily life. In general, lacking of lighting is very serious because many things and applications depend on lighting, including our system. Therefore, lighting failure detection is an important function in our system. We describe the situation and features when lighting fails in Section 5.4.1. In Section 5.4.2, we describe the method we propose for lighting failure detection.

### 5.4.1 Lighting Failure Features

Lighting influences the image captured by the camera. Actually, the intensity of an image represents the lighting situation in the environment. In other words, the change of the intensity of images represents the change of the lighting in the environment, and vice versa.

The intensity of an image captured by the camera in a normal situation usually turns out to be an ordinary value. On the contrary, the intensity of an image captured at a lighting failure situation usually becomes a lower value. The difference of the intensity is useful to judge the lighting situation. When lighting fails, the intensity will decrease suddenly. This causes a large difference between the previous intensity image and the present one. Figure 5.6 shows the two consecutive images of an ordinary situation and Figure 5.7 shows two consecutive images when the lighting fails.



(a)



(b)

Figure 5.6 The two consecutive images in ordinary situation.



(a)



(b)

Figure 5.7 The two consecutive images when the lighting fails. (a) The ordinary image. (b) The image of lighting failure.

## 5.4.2 Process for Lighting Failure Detection

Because the intensity of the image represents the environment changes obviously when lighting fails, an effective method to detect the change is to monitor the image continuously. A reasonable assumption is that the environment is ordinary at first. Thus, we capture the initial image and transform it into HSI color model by Equation 5.1. Then, we compute the histogram of the intensity of the image to find out the largest and the smallest values. These two values are usually the brightest and the darkest part of the image. However, because of the factor of the light and the camera, the image is usually over exposed. The brightest data are less significant. With similar

reasons, the darkest data usually represent the part where the light does not cover. We discard these two components and compute the remainder. The mean of the remainder values represents the intensity of this image and we denote it as  $Intensity_A$ .

Then, we capture an image from the camera again in the next time and use the same method to compute the intensity of the image. Thus, the mean of the intensity of the image is computed and denoted as  $Intensity_B$ , which represents the lighting situation in the environment at this time. The change of the intensity of two consecutive images comes from  $Intensity_A - Intensity_B$ . The lighting of most situations is not always the same everywhere in general, so the intensity of images is also not the same every time. It needs to tolerate some reasonable changes of the intensity of two continuous images.

However, when the lighting fails, the change of the intensity of two consecutive images is *obviously*, so we use a lot of images to obtain an appropriate value. This experimental value is used to judge whether the intensity changes in a tolerable range or not. Once the change of the intensity is large enough, it means that the lighting fails in the environment. We also use the experimental result to obtain another threshold value. Whenever the intensity is below the value, it means that the lighting fails. At the end of each cycle, we update the  $Intensity_A$  to be the new one  $Intensity_B$  for the next cycle of lighting failure detection. Thus, we can compare the two latest lighting situations in the environment. The complete fire detection process is described in the following algorithm and a detailed flowchart is illustrated in Figure 5.11 Some experience results are shown in Figure 5.8, Figure 5.9 and Figure 5.10.

**Algorithm 5.2** *Lighting failure detection.*

*Input:* A color image  $I$ .

*Output:* A warning message or nothing.

*Steps:*

- Step 1. Transform the image  $I$  in RGB color model to HSI color model by Equation 5.1. Represent the intensity as  $I_I$ .
- Step 2. Compute the histogram of  $I_I$  and discard the highest and the lowest values.
- Step 3. Compute the mean of the remaining intensity values and denote the result as  $Intensity_A$ .
- Step 4. Capture a new image of the latest environment.
- Step 5. Repeat Step 1 to Step 3 to compute the intensity of the latest image and denote the intensity as  $Intensity_B$ .
- Step 6. Compute the difference between  $Intensity_A$  and  $Intensity_B$ . If the difference is out of a tolerable range, then show a warning message.
- Step 7. If  $Intensity_B$  is too low, then show a warning message.
- Step 8. Update the  $Intensity_A$  to be the new  $Intensity_B$  for the next cycle of lighting failure detection..



(a)



(b)

Figure 5.8 The continuous images. (a) The initial situation. (b) The ordinary situation after (a). (c) The image of the lighting failure. (d) The continuous image after (c).

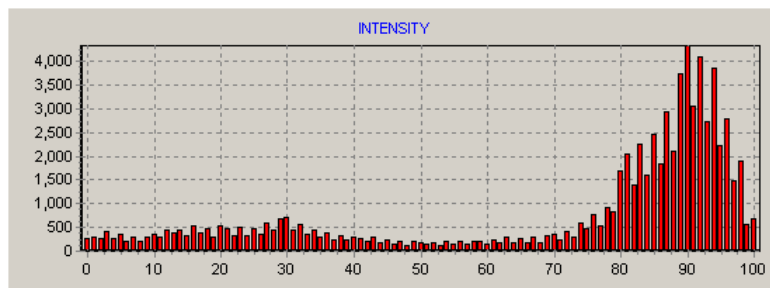


(c)

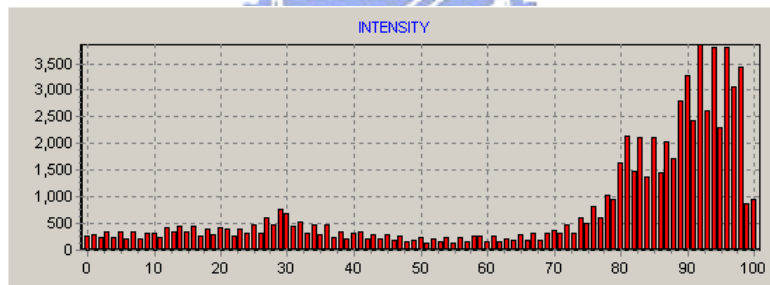


(d)

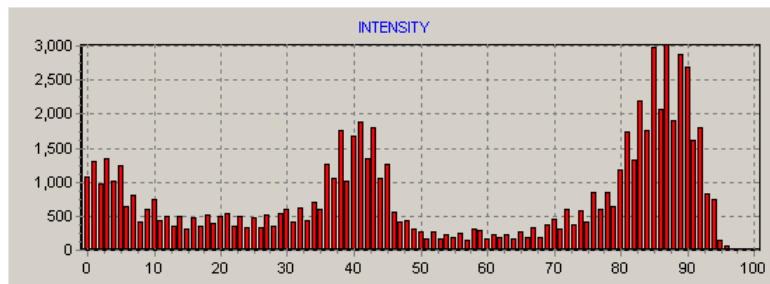
Figure 5.9 The continuous images. (a) The initial situation. (b) The ordinary situation after (a). (c) The image of the lighting failure. (d) The continuous image after (c).



(a)

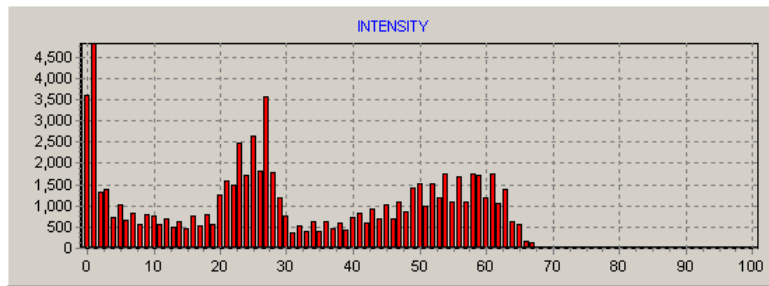


(b)



(c)

Figure 5.10 The intensity histograms respective to Figure 5.X. (a) The mean of the intensity is 0.73. (b) The mean of the intensity is 0.73. (c) The mean of the intensity is 0.55. (d) The mean of the intensity is 0.32.



(d)

Figure 5.10 The intensity histograms respective to Figure 5.X. (a) The mean of the intensity is 0.73. (b) The mean of the intensity is 0.73. (c) The mean of the intensity is 0.55. (d) The mean of the intensity is 0.32.

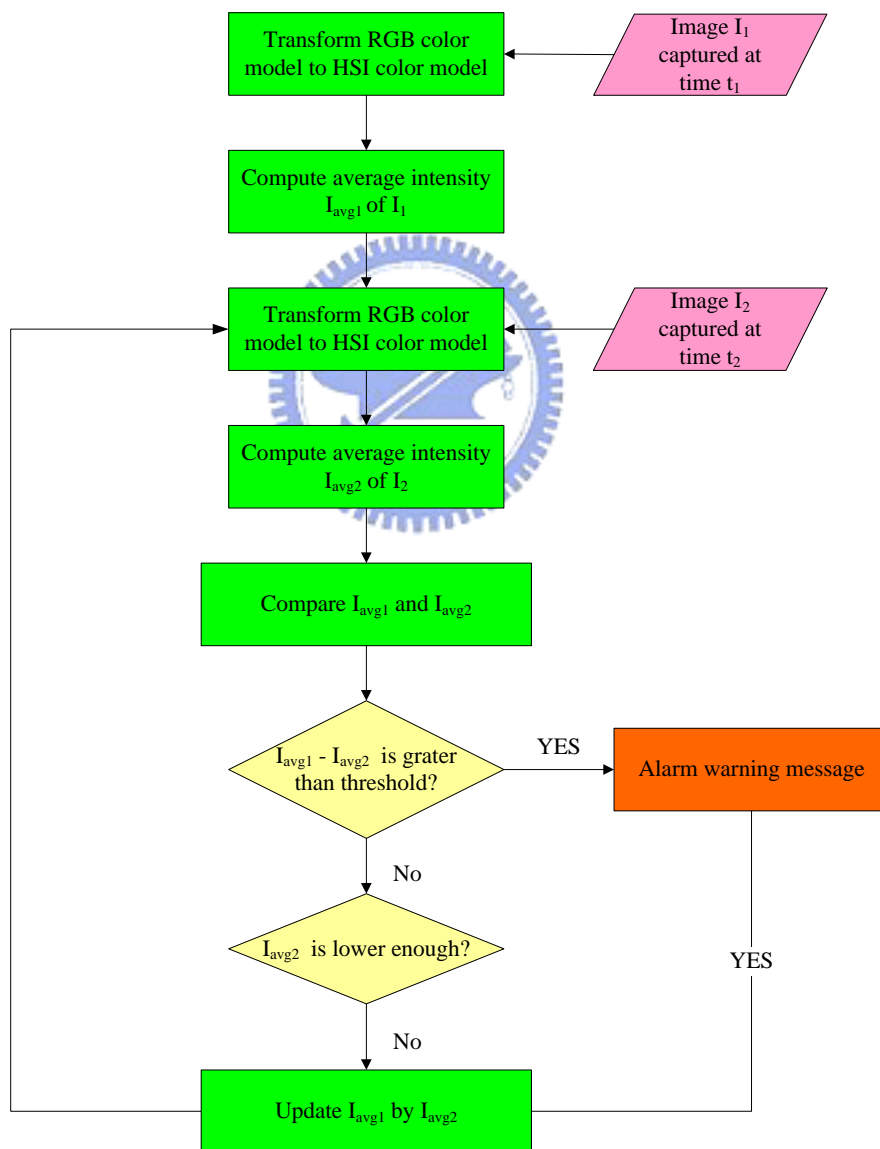


Figure 5.11 Flowchart of lighting failure detection.



# Chapter 6

## Security Monitoring of Paintings on Walls in Indoor Environments by Image Rectification Technique

### 6.1 Overview

In the past, a vehicle equipped with a fixed camera can only perform security monitoring of objects on the floor. But it is different to use a vehicle equipped with a PTZ camera. By means of the PTZ camera, security monitoring of objects is no longer restricted to those on floor. Thus, we take the advantage to monitor paintings on walls. Nevertheless, the vehicle still navigates on the floor, and the view of the camera is still from a low position. An image captured from a low position is always skewed due to perspective transformation. It is inconvenient to get correct information from a skewed image. Therefore, we have to rectify a skewed image to get more precise information of a painting for security monitoring. The problem of rectification is described in Section 6.2. Once we can rectify a skewed image to get some features of the painting, we need a method to detect a painting on the wall automatically in vehicle security surveillance. A method proposed in this study is described in Section 6.3.

## 6.2 Rectification of Images of Paintings on Walls

In this Section, we discuss the reason why an image should be rectified for security monitoring of a painting. In the past, we can only perform security monitoring of objects on floor. Nowadays, with a PTZ camera, security monitoring of objects at higher areas like paintings on walls is possible. However, security monitoring of an object not on floor is different from the same work in the past. We describe the problem met in this study in Section 3.2.1. And a method to overcome the problem is described in Section 3.2.2.

### 6.2.1 Difference Between Images of Paintings on Walls And Objects on Floors

Because the vehicle is small, the view of the camera on the vehicle is from a low position. In general, if we observe objects in the same height as the camera and face the camera, the image grabbed by the camera looks ordinarily. Once the camera pans or tilts an angle, the image does not look like an ordinary one. It is skewed like the example shown in Figure 6.1. This is a kind of distortion, called perspective transformation coming from perspective imaging. For instance, in Figure 6.1, a card with an original shape of a rectangle is no longer with a rectangular shape in the image.

In general, parallel lines on a scene plane are not parallel in the image but instead converge to a finite point, called vanish point. In this condition, we lose the real shape and the boundary ratio data of an object, which is a painting in this study.



Figure 6.1 A skewed image.

## 6.2.2 Rectification of Skewed Images

We need more actual shapes and side ratio information of a painting on a wall in this study, so a process to recover such an image is necessary. Because a planar projective transformation is a linear transformation on homogeneous 3-vectors, we can represent the transformation as  $x' = Hx$ , where  $x' = (x_1', x_2', x_3')^T$  and

$x = (x_1, x_2, x_3)^T$  are a pair of matching points, and  $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$  is the

homogeneous matrix for linear perspective transformation. Let the inhomogeneous coordinates of a pair of matching points  $x$  and  $x'$  in the world and in the image plane be  $(x, y)$  and  $(x', y')$ , respectively. The projective transformation can be written in inhomogeneous form as Equations 6.1 and 6.2 below:

$$\frac{x_1'}{x_3'} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}; \quad (6.1)$$

$$\frac{y'}{x_3'} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}. \quad (6.2)$$

Each point correspondence generates two equations for the elements of H, which after multiplying out are as Equations 6.3 and 6.4 below.

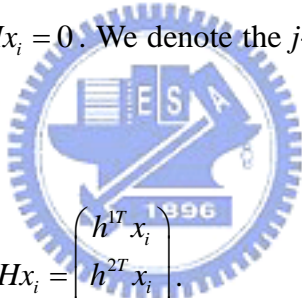
$$x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13}; \quad (6.3)$$

$$y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23}. \quad (6.4)$$

So four point correspondences lead to eight such linear equations in the entries of  $H$ , which are sufficient to solve for  $H$  up to an insignificant multiplicative factor. A method to solve  $H$  directly as a homogeneous vector is to turn the set into an inhomogeneous set of linear equations. But this method is slow and inefficient. In this study we use the direct linear transformation (DLT) [14] to obtain  $H$ .

At first, the equation  $x' = Hx$  involves homogeneous vectors. It means the 3-vectors  $x'$  and  $Hx_i$  are not equal. They have the same direction but may differ in magnitude by a non-zero scale factor. The equation can be expressed in terms of the vector cross product as  $x_i \times Hx_i = 0$ . We denote the  $j$ -th row of the matrix  $H$  as  $h^{jT}$ ,

then we obtain



$$Hx_i = \begin{pmatrix} h^{1T} x_i \\ h^{2T} x_i \\ h^{3T} x_i \end{pmatrix}. \quad (6.5)$$

By writing  $x_i' = (\alpha'_i, \beta'_i, \gamma'_i)$ , the cross product may then be given explicitly as

$$x_i \times Hx_i = \begin{pmatrix} \beta'_i h^{3T} x_i - \gamma'_i h^{2T} x_i \\ \gamma'_i h^{3T} x_i - \alpha'_i h^{2T} x_i \\ \alpha'_i h^{3T} x_i - \beta'_i h^{2T} x_i \end{pmatrix}. \quad (6.6)$$

Since  $h^{jT} x_i = x_i^T h^j$  for  $j=1, 2, 3$ , this gives a set of three equations in the entries of  $H$  in the following form:

$$\begin{bmatrix} 0^T & -\gamma'_i x_i^T & -\beta'_i x_i^T \\ -\gamma'_i x_i^T & 0^T & -\alpha'_i x_i^T \\ -\beta'_i x_i^T & -\alpha'_i x_i^T & 0^T \end{bmatrix} \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix} = 0. \quad (6.7)$$

These equations have the form  $A_i h = 0$ , where  $A_i$  is a  $3 \times 9$  matrix, and  $h$  is a 9-vector made up of the entries of the matrix  $H$ . The relationship between  $h$  and  $H$  is described as follows:

$$h = \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix}; \quad (6.8)$$

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (6.9)$$

with  $h_{ij}$  is the  $j$ -th element of  $h^i$ .

Each point correspondence provides two independent equations in the entries of  $H$ . Give a set of four such point correspondences, we obtain a set of equations  $Ah=0$ , where  $A$  is the matrix of the equation coefficients from the matrix rows  $A_i$  which is obtained from each correspondence, and  $h$  is the vector of the unknown entries of  $H$ . We seek a non-zero solution  $h$ , since the obvious solution  $h=0$  is useless. Finally,  $H$  is in general determined up to scale, so the solution  $h$  gives the required  $H$ . A scale may be arbitrarily chosen for  $h$  by a requirement on its norm such as  $\|h\|=1$ . The detailed algorithm is in the following.

**Algorithm 6.1** *The direct linear transform for  $H$ .*

*Input:* Four 2D to 2D point correspondences  $\{x_i \leftrightarrow x_i'\}$ .

*Output:* The homogeneous matrix  $H$ .

*Steps:*

- Step 1. For each correspondence  $x_i \leftrightarrow x_i'$ , compute the matrix  $A_i$  by Equation 6.7. Only the first two rows need be used in general.
- Step 2. Assemble the  $n$   $2 \times 9$  matrices  $A_i$  into a single  $8 \times 9$  matrix  $A$ .
- Step 3. Obtain the singular value decomposition of  $A$ . The unit singular vector corresponding to the smallest singular value is the solution  $h$ .
- Step 4. Determine the matrix  $H$  from  $h$  by Equations 6.8 and 6.9.

## **6.3 Security Monitoring of Paintings on Walls by Image Analysis**

In this section, we propose a method for security monitoring of paintings on walls. For the purpose of monitoring, the vehicle needs to learn the navigation node of the painting and the painting features. We propose the method of learning paintings in Section 6.3.2. After the learning stage, we propose a method for the vehicle to accomplish security monitoring of paintings automatically. We describe the detail in Section 6.3.3. Before we learn the painting features, we need to introduce the features of painting images. We describe them in Section 6.3.1 at first.

### **6.3.1 Painting Image Features**

In most paintings, contents of paintings are not features. Figure 6.2 shows a painting on a wall in general. However, most paintings are almost rectangular and

bordered. These two features are helpful to detect a painting and find point correspondences in rectification. Although borders can help to detect a painting, we also need content features to judge whether a painting still existing or not. In order to compare the similarity of the two images, we save a rectified image of a painting in learning. So we can get any data from the original image of the painting. Specifically, we use the gray image of the painting as the saved painting data.



Figure 6.2 A painting on a wall image.

### 6.3.2 Learning Process

In order to accomplish security monitoring of paintings on walls, we must perform the learning process of paintings at first. We need to record the vehicle location, the PTZ camera position, and the rough area about the painting in the learning process. Before the learning process, we describe the proposed method to rectify a skewed image at first.

In order to rectify a skewed image, we need the transformation matrix  $H$  at first. The determination of  $H$  by Algorithm 6.1 needs at least four point correspondences. Furthermore, as the point correspondences are more, the resulting  $H$  is more precise. Assume that we know the rough area of the painting in the image and the wall is plain. This can be accomplished by manual control in advance. Then we apply the region

growing algorithm from the border of the rectangle to find the outline of the painting. After the region growing process, we apply the Hough transform to find the four edges of the painting. We can represent each edge by  $u + bv + c = 0$  by Equations 3.27 and 3.28. By the equation form, the four intersections of the four edge equations are obtained. We denote the four intersections of the left-top, the right-top, the left-bottom, and the right-bottom of a painting as  $P_1, P_2, P_3,$  and  $P_4,$  respectively. We also define four points corresponding to  $P_1, P_2, P_3,$  and  $P_4$  after rectification as  $P_1', P_2', P_3',$  and  $P_4',$  respectively. Because the outline of a painting is usually a rectangle, the painting appearing in the image is also a rectangle after rectification. This provides the following features to help to determine the point correspondences.

- (1)  $\overline{P_1'P_2'}$  is perpendicular to  $\overline{P_1'P_3'}$ .
- (2)  $\overline{P_1'P_2'}$  is parallel to  $\overline{P_3'P_4'}$ .
- (3)  $\overline{P_1'P_3'}$  is parallel to  $\overline{P_2'P_4'}$ .
- (4) The distance of  $\overline{P_1'P_2'}$  equals the distance of  $\overline{P_3'P_4'}$ .
- (5) The distance of  $\overline{P_1'P_3'}$  equals the distance of  $\overline{P_2'P_4'}$ .

We assume that the  $P_1'$  equals the  $P_1$  in the ICS and the coordinates of  $P_1$  is  $(u_1, v_1)$ . We also take the distance of  $\overline{P_1P_2}$  and  $\overline{P_1P_3}$  as the distance of  $\overline{P_1'P_2'}$  and  $\overline{P_1'P_3'}$ , respectively. Thus, we can find the coordinates of  $P_1'$  through  $P_4'$  in the image coordinate system by the following equations:

$$P_1' = P_1 \quad (6.11)$$

$$P_2' = P_1' + (D_{12}, 0) = (u_1 + D_{12}, v_1) \quad (6.12)$$

$$P_3' = P_1' + (0, D_{13}) = (u_1, v_1 + D_{13}) \quad (6.13)$$



$$P_4' = P_1' + (D_{12}, D_{13}) = (u_1 + D_{12}, v_1 + D_{13}) \quad (6.14)$$

where  $D_{12}$  and  $D_{13}$  are the distances of  $\overline{P_1P_2}$  and of  $\overline{P_1P_3}$ , respectively.

Now we obtain the four point correspondences, so we can find the matrix  $H$  by Algorithm 6.1. A flowchart is illustrated in Figure 6.3, and the detail is described as an algorithm in the following.

**Algorithm 6.2** *Computation of the matrix  $H$ .*

*Input:* A color image  $I$  and the rectangle including the painting.

*Output:* The transformation matrix  $H$ .

*Steps:*

- Step 1. Apply region growing from the border of the rectangle in the rectangle to find the outline of the painting.
- Step 2. Apply the Hough transform to detect the four edges of the painting.
- Step 3. Transform the edge lines into the form  $u + bv + c = 0$  by Equations 3.27 and 3.28.
- Step 4. Compute the four intersections  $P_1, P_2, P_3,$  and  $P_4$  of the four edge line equations.
- Step 5. Compute the four corresponding points  $P_1', P_2', P_3',$  and  $P_4'$  by Equations 6.12 to 6.14.
- Step 6. Use the four point correspondences to find the transformation matrix  $H$  by Algorithm 6.1.

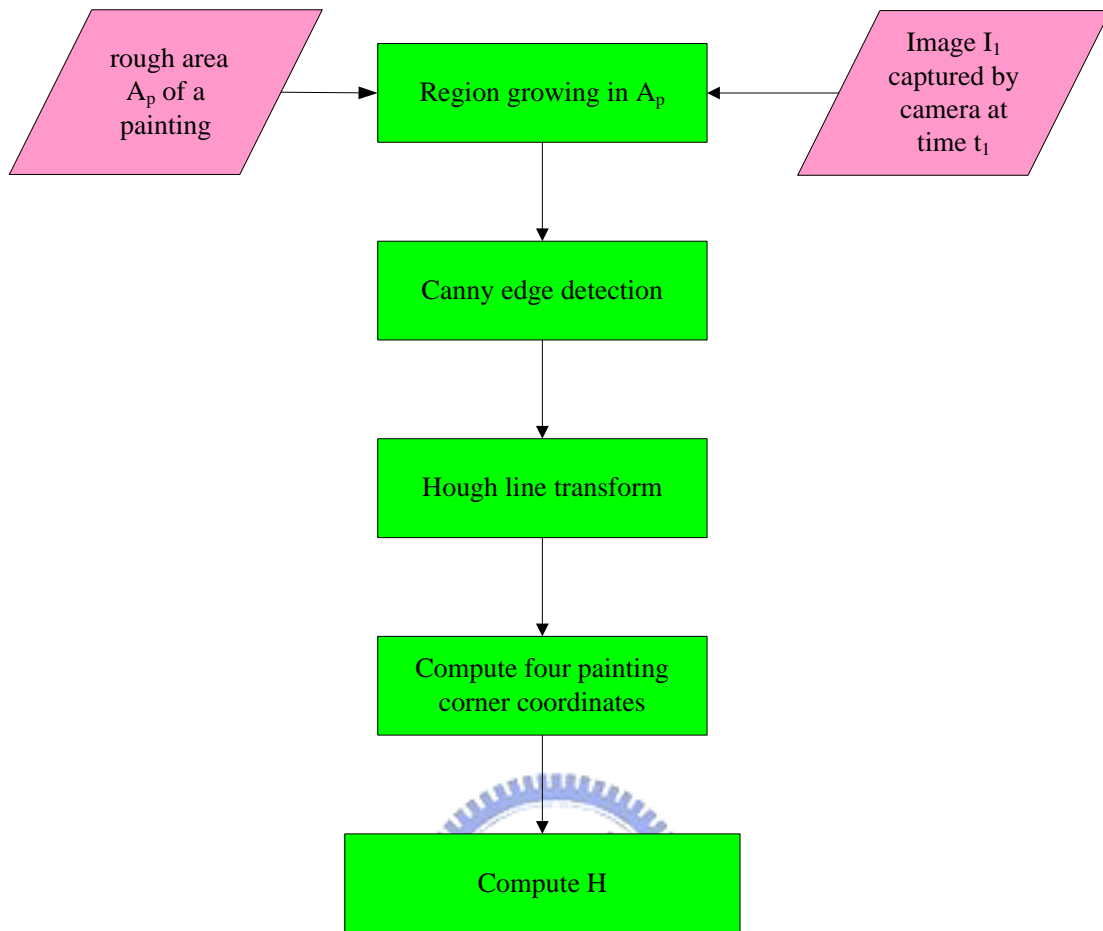


Figure 6.3 The flowchart of computation of the matrix  $H$ .

As long as we have the transformation matrix  $H$ , we can obtain the rectified painting image by  $H$ . At the same time, we also know the painting position in the rectified image by  $P_1'$ ,  $P_2'$ ,  $P_3'$ , and  $P_4'$ , so we can save the rectified painting image for security monitoring. At the beginning of the learning process, we move the vehicle to a monitoring position and move the PTZ camera toward the painting. Then, we use the mouse connected to the PC to drag a rectangle to cover the painting which appears in the image. Now, we use the transformation matrix  $H$  obtained by Algorithm 6.2 to rectify the skewed image. Based on  $P_1'$ ,  $P_2'$ ,  $P_3'$ , and  $P_4'$ , we know the rectified painting in the rectified image, so we can save the image for the security monitoring. We also save the vehicle location, the PTZ position, and the rough painting position in the image. A flowchart is illustrated in Figure 6.4, and the detail is described in the

following. Some experimental results are shown in Figure 6.5.

**Algorithm 6.3** *Learning of a painting..*

*Input:* A color image  $I$  including a painting.

*Output:* The rectified painting image, the camera position, the vehicle location, and the painting position in the image roughly.

*Steps:*

- Step 1. Move the vehicle to the painting position.
- Step 2. Move the PTZ camera toward the painting.
- Step 3. Drag a rectangle on the image including the painting.
- Step 4. Use Algorithm 6.2 to find the transformation matrix  $H$ .
- Step 5. Rectify the image by  $H$ .
- Step 6. Save the rectified painting in the rectified image.
- Step 7. Save the vehicle location, the PTZ position, and the rough painting position in the image.

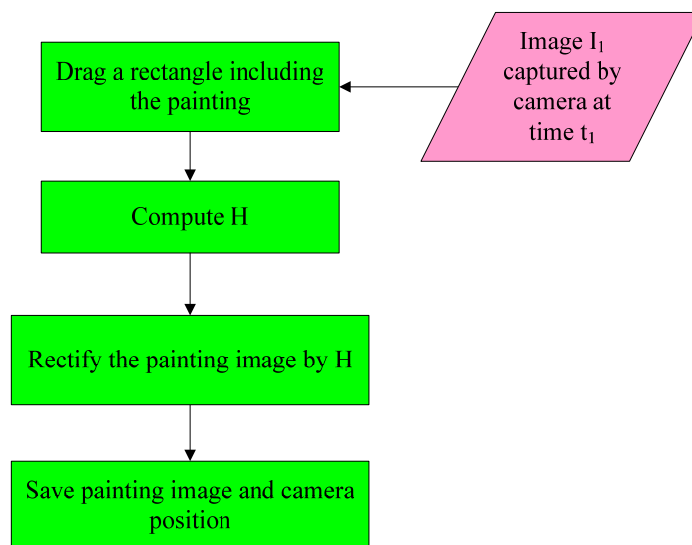


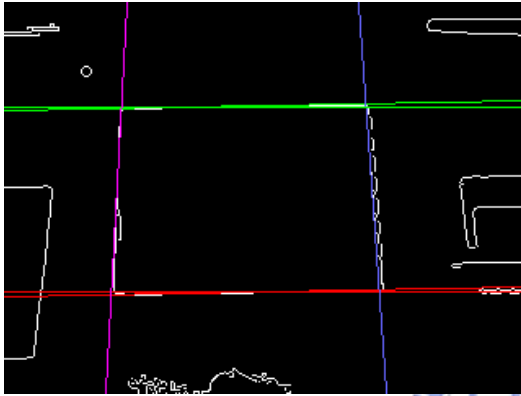
Figure 6.4 The flowchart of the learning painting process.



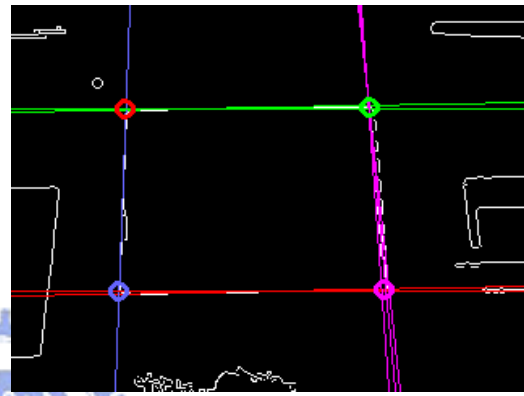
(a)



(b)



(c)



(d)



(e)



(f)

Figure 6.5 The experimental result of learning a painting. (a) A painting on a wall image. (b) A rectangle including the painting. (c) The four borders of the painting. (d) The four intersections computed from the four borders. (e) The four point correspondences. (f) The rectified painting.

### 6.3.3 Painting Security Monitoring Process

After the learning process, we have a set of data including the navigation nodes and security monitoring paintings. We take the navigation data from the learning process to guide the vehicle to the certain position for painting security monitoring. Before we check whether the painting is the same as the one in the learning process or not, we should detect whether the painting exists or not.

Because we have stored the rough area of the painting in the image, we can utilize the area to detect the painting first. For the purpose of error tolerance for the image error due to the tolerable error of the vehicle position, we enlarge the area to detect the painting. Within the area, we use the Hough transform to detect the borders of the painting and represent each border by Equation 3.25. Then, we group the borders into four groups by the coefficients  $\alpha$  and  $\gamma$ . If the four groups of borders cannot be detected successfully in certain times with different thresholds of the Hough transform, we may claim the painting is not there any more. Then, we show a warning message for the painting mismatch. The detail is described in the following as an algorithm.

**Algorithm 6.4** *Detection of monitored painting.*

*Input:* A color image  $I$  and the learned painting area in the image.

*Output:* A Boolean value, true or false.

*Steps:*

- Step 1. Enlarge the learned painting area in the image by an appropriate ratio.
- Step 2. Apply the Hough transform to detect the four borders of the painting.
- Step 3. Group the borders into four groups by  $\alpha$  and  $\gamma$ .
- Step 4. If the four groups of borders are detected successfully, return true.

Step 5. Repeat Steps 2 through 4 with different thresholds of the Hough transform for five times.

Step 6. After Step 5, if the four groups of borders still cannot be found, we return false. This represents that the painting is not detected successfully.

Since we have detected the painting in the image, we need to compare the painting with the learned one. At first, we use Algorithm 6.2 to find the  $H$  to rectify the painting image. Then we extract the rectified painting in the image. Because the sizes of the rectified painting and that of the learned painting may be not the same, we apply size normalization to obtain the two paintings with the same size. We rescale the size of the larger one to the smaller size. Thus, we have the two paintings with the same size. In order to compare the similarity of the two paintings, we transform the two paintings into gray images first. Before the comparison, we define a *block* in the image as a square region of pixels, which is the unit of the image and takes the mean of the gray values of the square regions as a color value. A *block image* is composed of these blocks segmented from the original image. Then we can segment the two painting images into two block image as  $S_a$  and  $S_b$ . Finally a new block image  $S_d$  representing the difference of the  $S_a$  and  $S_b$  is obtained by overlapping  $S_a$  and  $S_b$ . Because the central part of the image is more important, so we discard the border of the block set  $S_d$ . Then, the number of the remainders which are over a threshold is recorded as  $N_{diff}$ . The value  $N_{diff}$  represents the obvious difference between the two painting images. Then, we can compute the difference ratio by  $\frac{N_{diff}}{BlockSize}$  where  $BlockSize$  is the number of blocks of  $S_d$ . The detail is described in the following and a flowchart is illustrated in Figure 6.6. Some experimental results are shown in Figure 6.7.

**Algorithm 6.5** *Painting matching process.*

*Input:* A painting image  $I$  and a learned painting image  $I_l$ .

*Output:* A Boolean value, true or false.

*Steps:*

- Step 1. Use Algorithm 6.2 and the learned painting area to find  $H$  and obtain the rectified painting image  $I_r$ .
- Step 2. Normalize the size of the two images  $I_r$  and  $I_l$ .
- Step 3. Convert  $I_r$  and  $I_l$  into gray-level images  $I'_r$  and  $I'_l$ .
- Step 4. Segment the images  $I'_r$  and  $I'_l$  into block image  $S_a$  and  $S_b$ .
- Step 5. Overlap  $S_a$  and  $S_b$  to obtain  $S_d$ .
- Step 6. Count the number  $N_{diff}$  of blocks with obvious difference,.
- Step 7. Compute the difference ratio by  $\frac{N_{diff}}{BlockSize}$
- Step 8. If the ratio is too large, return false.

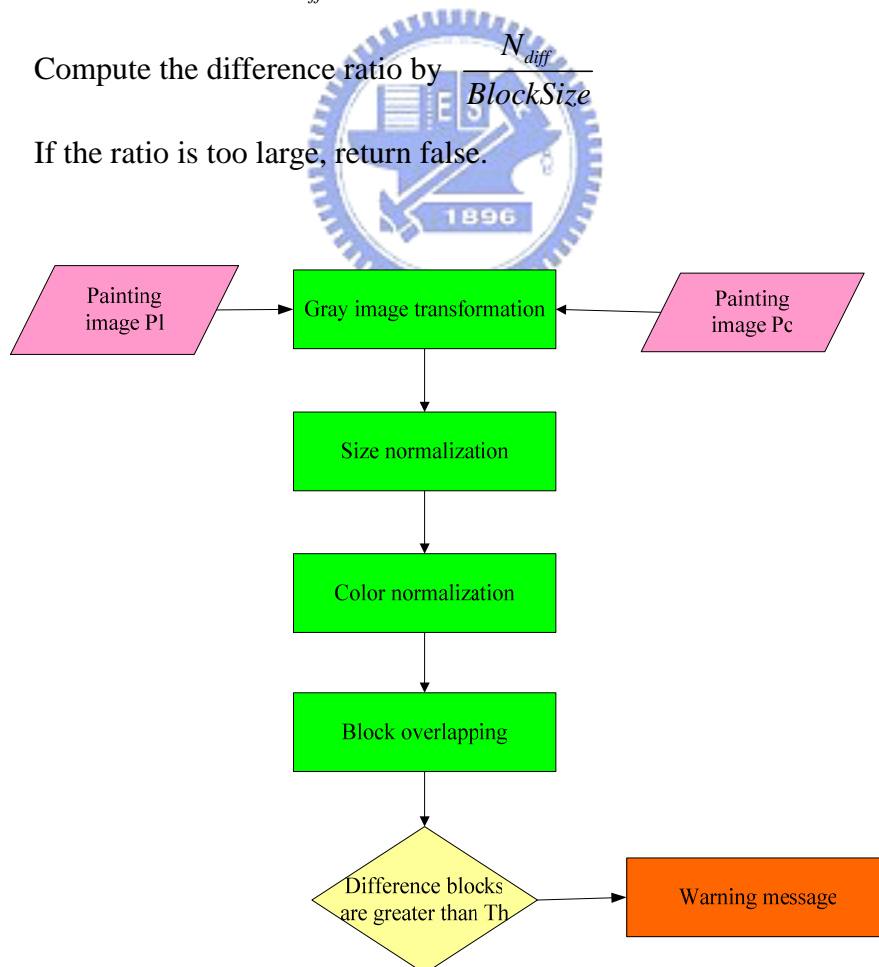


Figure 6.6 The flowchart of the painting matching process.

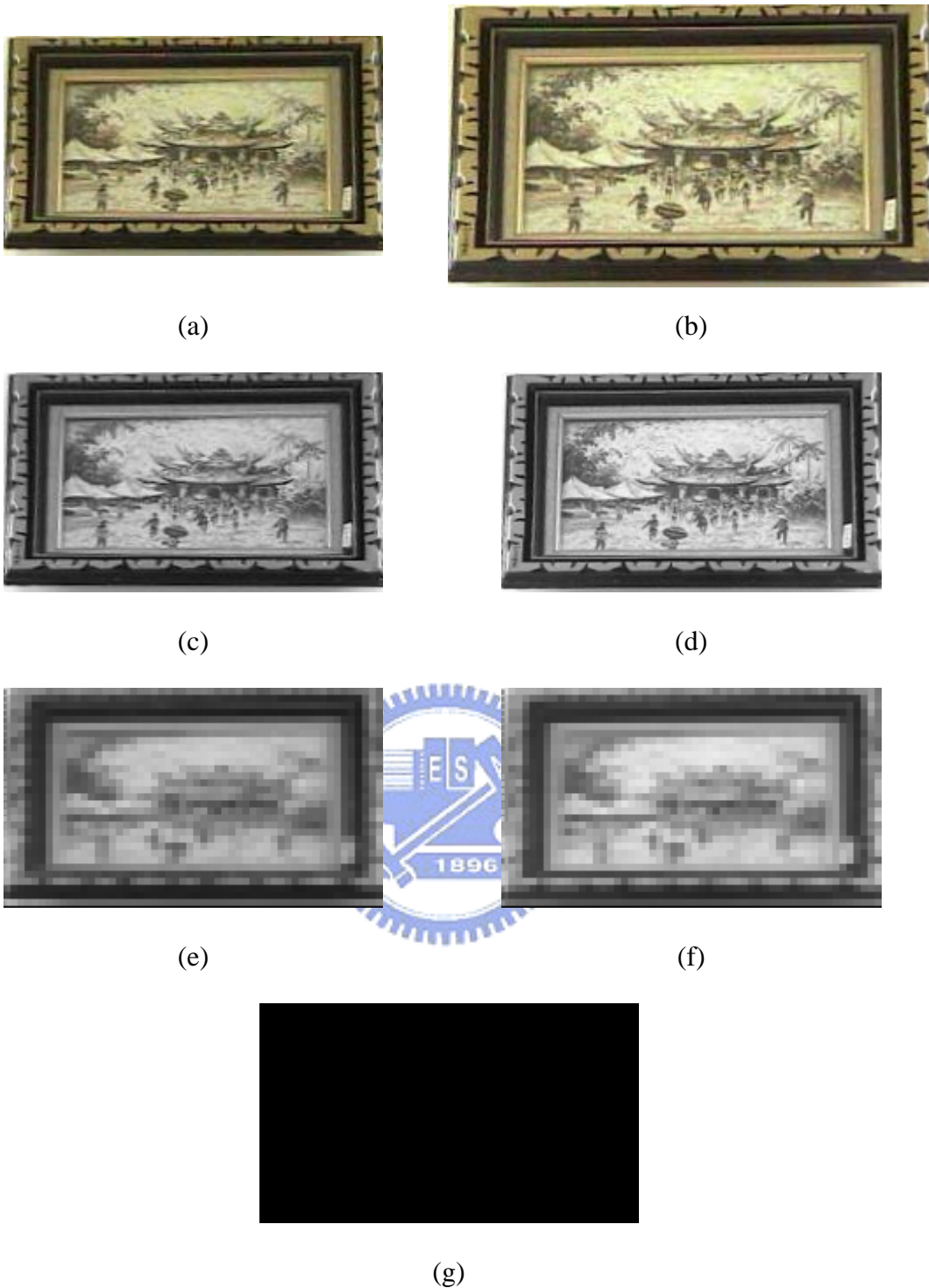


Figure 6.7 The experimental results of the painting matching process. (a) and (b) are the two different rectified painting image of the same painting. (c) and (d) are the normalized image in gray-level of (a) and (b) respectively. (e) and (f) are the block image of (c) and (d) respectively. (g) is the overlapping result of (e) and (f).





Figure 6.8 The experimental results of the painting matching process of two different paintings. (a) and (b) are the two different rectified painting image of the two different paintings. (c) and (d) are the normalized image in gray-level of (a) and (b) respectively. (e) and (f) are the block image of (c) and (d) respectively. (g) is the overlapping result of (e) and (f).

Now, we possess the ability to detect whether a painting exists or not and to match a painting with the learned one, so we can perform a painting security monitoring process. At first, we move the vehicle to the monitoring node and move the camera to face the painting according to the learned data. Then we use Algorithm 6.4 to detect whether a painting exists or not. If a painting is not there, we announce a warning message. Once we detect a painting, we use Algorithm 6.5 to match the painting with the learned one. If the painting matching process returns a false value, we announce a warning message about the mismatch of the painting. The detail is described in the following as algorithm and a corresponding flowchart is illustrated in Figure 6.9.

**Algorithm 6.6** *Painting security monitoring process.*

*Input:* A color image  $I$  and learned data  $P_d$ .

*Output:* A warning message or nothing.

*Steps:*

- Step 1. Move the vehicle to the security monitoring node according to the learned navigation data.
- Step 2. Move the camera to face the painting.
- Step 3. Use Algorithm 6.4 to detect a painting. If the result is false, announce a warning message about painting mismatch.
- Step 4. Use Algorithm 6.5 to decide whether the painting matches the learned painting or not. If the result is false, announce a warning message about painting mismatch.
- Step 5. If the border cannot be found, announce a warning message.



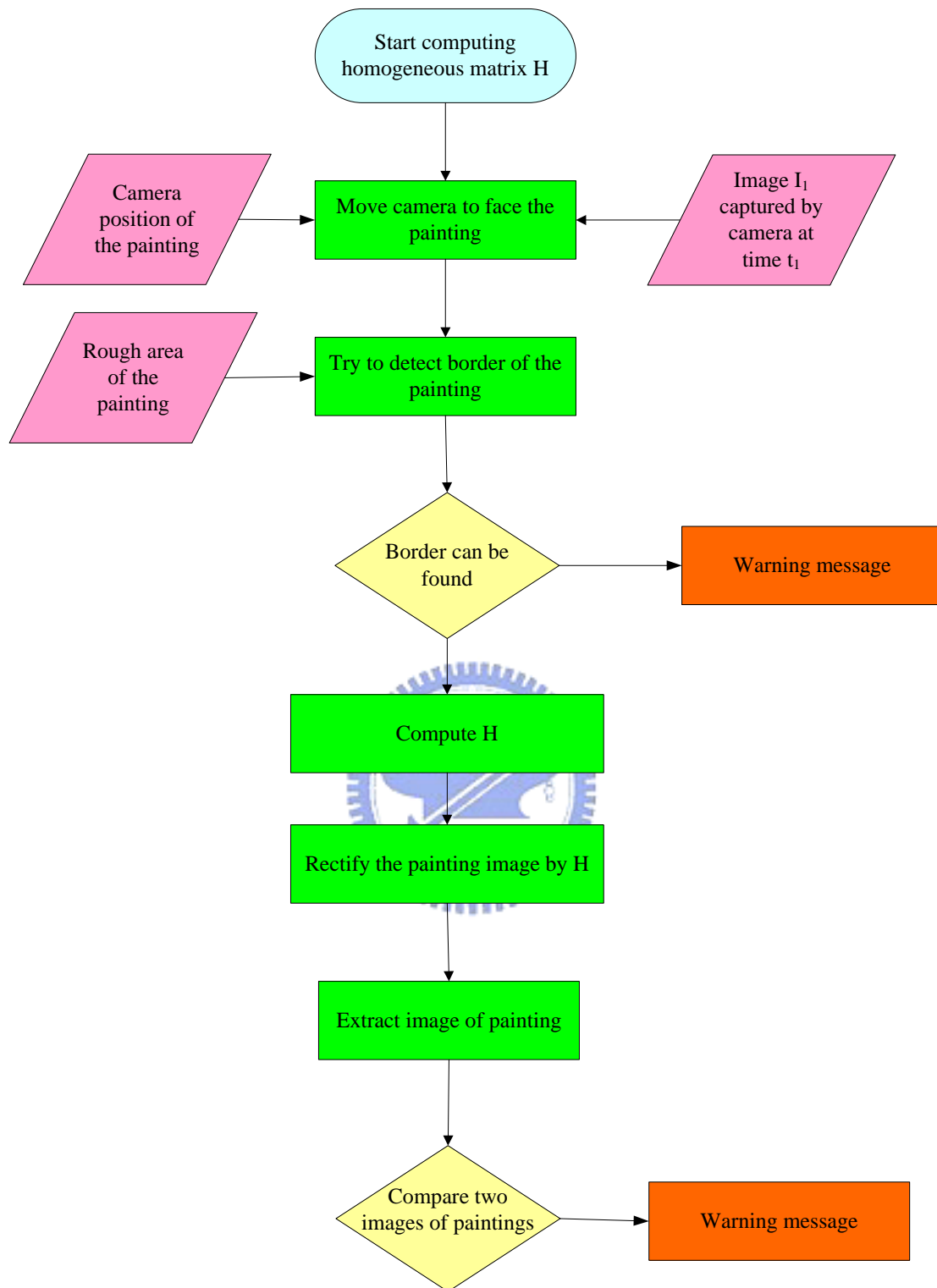


Figure 6.9 The flowchart of the security monitoring process.

# Chapter 7

## Experimental Results and Discussion

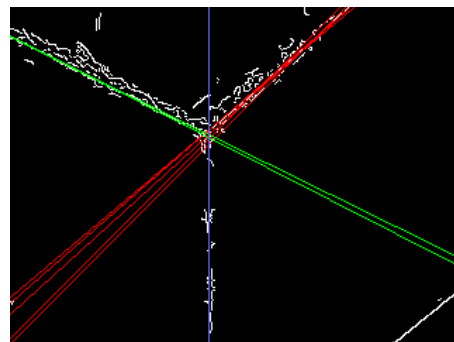
### 7.1 Experimental Results

In this section, we will show some experimental results of the proposed security patrolling system with danger condition monitoring.

At first, a user controls the vehicle to learn a path and some monitored paintings in walls. Whenever the vehicle arrives at a position, a user controls the camera to observe a house corner for vehicle location estimation. This action will update the vehicle location in the world coordinate system, and the system will then record the parameters of the camera in order to calibrate the vehicle location to move it to the planned position in navigation. An experimental result of learning is shown in Figure 7.1. Figure 7.1(a) shows a corner used in the learning process. Figure 7.1(b) illustrates the result of the house corner edge lines used to estimate the vehicle location. Figure 7.1(c) shows a painting on wall. Figure 7.1(d) shows the saved painting.



(a)



(b)

Figure 7.1 The used corner and learned image.



(c)



(d)

Figure 7.1 The used corner image and the learned image.

After the learning process, a navigation map is created. An illustration of the learned data, the navigation map and the actual navigation paths from the experiment is shown in Figure 7.2.

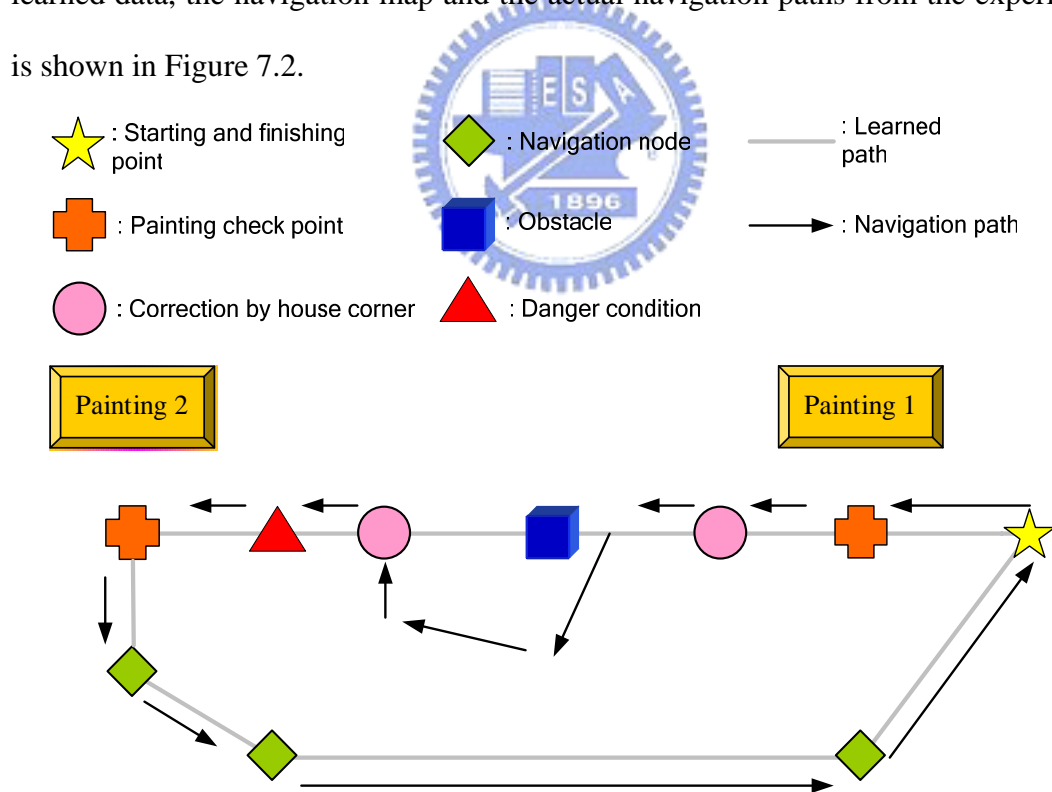


Figure 7.2 An illustration of learned data and navigation path.

The vehicle starts security patrolling by the created map. The navigation process is shown in Figure 7.3. Whenever the vehicle arrives at a node where the vehicle has

calibrated the location in the learning stage, it performs the same action to correct the vehicle location. Once the vehicle arrives at a learned painting node, it performs the security monitoring of the painting. If the vehicle detects an obstacle, it also avoids it.



(a)



(b)



(c)



(d)



(e)



(f)

Figure 7.3 A navigation process. (a) and (b) are ordinary situations. (c) and (d) are obstacle avoidance. (e) and (f) show the vehicle monitoring a painting.

Figure 7.3 (c) shows that the vehicle detects an obstacle and Figure 7.3 (d) shows that the vehicle changes the navigation path to avoid the obstacle. Figure 7.3 (e) shows that the vehicle uses the house corner to estimate the vehicle location by uplooking at a house corner with the camera. Figure 7.3 (f) shows that the vehicle detects the painting on the wall.

Some experimental results showing the vehicle action of detecting an obstacle, of detecting lighting failure or fire is shown in Figure 7.4.

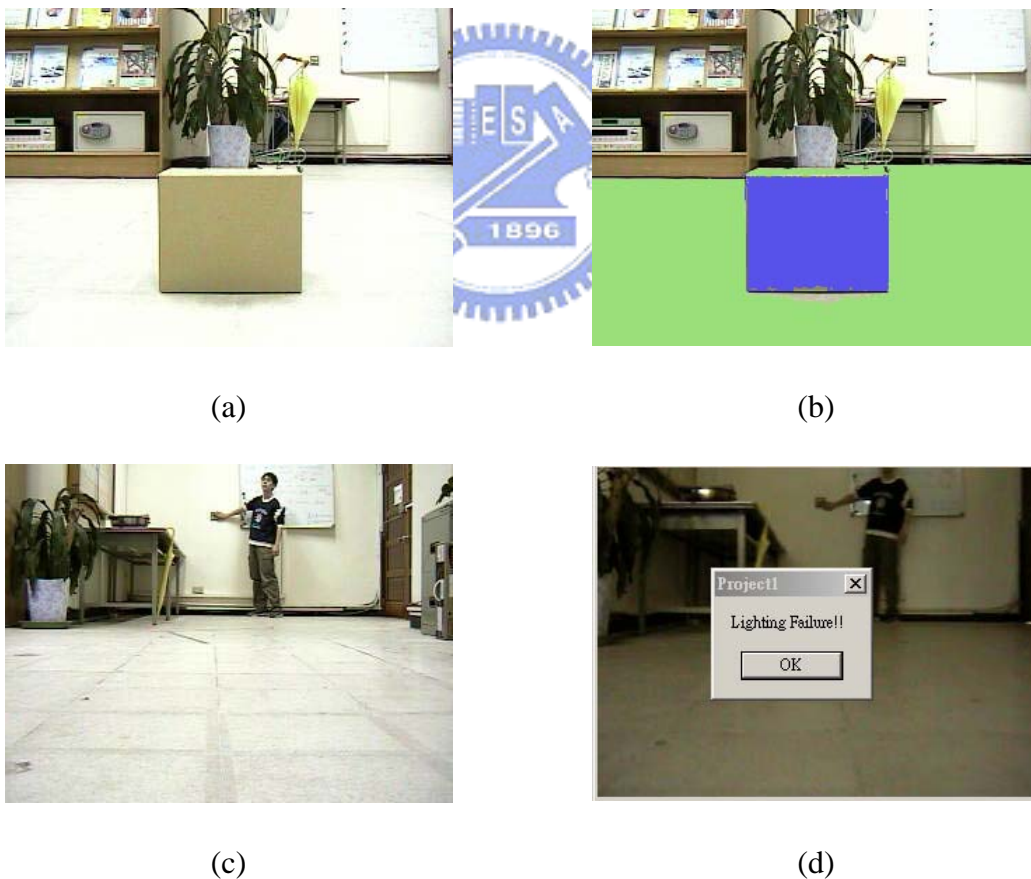
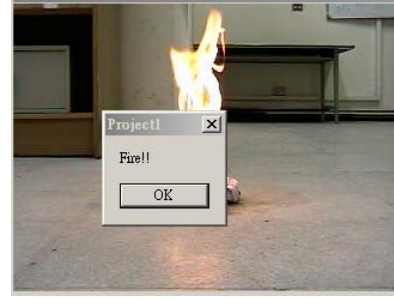


Figure 7.4 The experimental result of obstacle avoidance and detection of lighting failure and fire.



(e)



(f)

Figure 7.4 The experimental result of obstacle avoidance and detection of lighting failure and fire.

## 7.2 Discussion

We discover certain problems by analyzing the experimental results. And the problems are described as follows.

- (1) The result of using a house corner to estimate the vehicle location may exist error. This is because that the texture of the ceiling possesses a corner-like pattern and this pattern is nearby the house corner. Thus, the corner in the image is not identical, so the coefficients of the edge equations exist error. The vehicle location estimation also possesses the error.
- (2) The fire detection method may occur mistake due to the brightness of the environment is too high. This causes the intensity of the whole image is very high and the detection of the fire becomes worse. The fire and the background is difficult to separate.
- (3) That the floor has to be flat is a constraint of our system. An obstacle avoidance method used in this study needs to know the floor area. Otherwise, it is difficult to separate the floor and other things on the floor. The distance between the obstacle and the vehicle is also difficult to obtain.



# Chapter 8

## Conclusions and Suggestions for Future Works

### 8.1 Conclusions

In this study, several vision-based techniques and methods have been proposed and integrated into an autonomous vehicle system for security patrolling in the indoor environment with danger condition and on-wall painting monitoring capabilities.

At first, a vehicle guidance method is proposed, by which the vehicle follows the nodes in a learned map to patrol in indoor environment. The house corner is used to estimate the vehicle location to prevent mechanic error accumulation in navigation. Next, some danger condition detection methods are proposed for indoor environments, including obstacle avoidance, fire detection, and lighting failure detection. The system analyzes images continuously for detection of these danger conditions. In the proposed obstacle detection method, once the vehicle detects an obstacle on the floor, a new navigation node is inserted into the map to form a new path for obstacle avoidance. On the other hand, the HSI color model is used to deal with the entire image for fire and lighting failure detection because the fire and the lighting failure conditions possess obvious features in this model.

In addition, we have proposed a method for security monitoring of paintings on walls. We use a PTZ camera to monitor a painting on a wall. Because a painting on a wall is not the same with an object on the floor, we have proposed a method to rectify a painting image. By our method, we can detect a painting in the image and rectify the

skewed image to obtain a rectified painting image, which possesses correct image side ratio and color distribution. We have also proposed a method for painting matching. We convert a gray-level painting image into a block set and overlap two block sets of two different painting images to judge whether the painting images match or not. The experimental results have revealed the feasibility of the proposed system.

## 8.2 Suggestions for Future Works

The proposed strategies and methods, as mentioned previously, have been implemented on a vehicle system with a robot arm. Several suggestions and related interesting issues are worth further investigation in the future. We state them as follows.

- (1) Using an omni-directional camera for taking wider-view images of the environment.
- (2) Design a friendlier user-machine interface and simplifying the learning strategy.
- (3) Improve the painting detection process to be completely automatic without any human involvement.
- (4) Adding the capability of transmitting warning messages from the system to a user's cell phone or electronic mail address to warn a user immediately.
- (5) Adding the capability to detect more danger conditions.
- (6) Using more features instead of color feature to detect fire, such as smoke.

# References

- [1] C. C. Lai, "A Study on Automatic Indoor Navigation Techniques for Vision-Based Mini-Vehicle with Off-Line Environment Learning Capability," *Master Thesis*, Department of Computer and Information Science, National Chiao Tung University, Taiwan, June, 2003.
- [2] P. L. Li, "Path Learning, Planning, and Guidance for ALV Navigation Inside Buildings," *Master Thesis*, Department of Computer and Information Science, National Chiao Tung University, Taiwan, June, 1997.
- [3] Y. C. Chen and W. H. Tsai, "Vision-based autonomous vehicle navigation in complicated room environments with collision avoidance capability by simple learning and fuzzy guidance techniques," *Proceedings of 2004 Conference on Computer Vision, Graphics and Image Processing*, Hualien, Taiwan, Republic of China, August, 2004.
- [4] R. C. Liu, "Security Patrolling in Building Corridors by Multiple-Camera Computer Vision and Automatic Vehicle Navigation Techniques," *Master Thesis*, Department of Computer and Information Science, National Chiao Tung University, Taiwan, June, 2001.
- [5] M. C. Chen and W. H. Tsai "Vision-based security patrolling in indoor environments using autonomous vehicles," *Proceedings of 2005 Conference on Computer Vision, Graphics and Image Processing*, Taipei, Taiwan, Republic of China, August, 2005.
- [6] I. Fukui, "TV image processing to determine the position of a robot vehicle," *Pattern Recognition*, vol. 14, 1981, pp. 101-109.
- [7] M. J. Magee and J. K. Aggarwal, "Determining the position of a robot using a single calibration object," *Proceedings of IEEE Conference on Robotics*, Atlanta,

- Georgia, May, 1984, pp. 57-62.
- [8] H. L. Chou and W. H. Tsai “A new approach to robot location by house corners,” *Pattern Recognition*, vol. 19, 1986, pp. 439-451.
- [9] J. W. Courtney and J. K. Aggarwal, “Robot guidance using computer vision,” *Pattern Recognition*, vol. 17, no 5, 1984, pp. 585-592.
- [10] J. W. Courtney and J. K. Aggarwal, “Robot guidance using computer vision,” *Proceedings of IEEE Conference on Trends and Application, Automating Intelligent Behavior, Applications and Frontiers*. Gaithersberg, MD, May, 1983.
- [11] C. H. Ku and W. H. Tsai, “Obstacle avoidance in person following for vision-based autonomous land vehicle guidance using vehicle location estimation and quadratic pattern classifier,” *IEEE Transactions on Industrial Electronics*, vol. 48, no. 1, 2001, pp. 205-215.
- [12] Y. C. Chen and W. H. Tsai, “Vision-based autonomous vehicle navigation in complicated room environments with collision avoidance capability by simple learning and fuzzy guidance techniques,” *Proceedings of 2004 Conference on Computer Vision, Graphics and Image Processing*, Hualien, Taiwan, Republic of China, August, 2004.
- [13] R. C. Gonzalez and R. Z. Woods, *Digital Image Processing*, second edition, Prentice-Hall Inc., 2002.
- [14] Richard Hartley and Andrew Zisserman, *Multiple view Geometry in computer vision*, Cambridge University Press, 2000.